Contributions aux Modèles de Markov Cachés : métaheuristiques d'apprentissage, nouveaux modèles et visualisation de dissimilarité

Sébastien Aupetit

UNIVERSITÉ FRANÇOIS-RABELAIS
TOURS

École Doctorale : Santé, Sciences
et Technologies

Année universitaire : 2004-2005

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE TOURS

Discipline : Informatique

présentée et soutenue publiquement

par :
Sébastien AUPETIT
le 30 novembre 2005

CONTRIBUTIONS AUX MODÈLES DE MARKOV CACHÉS : MÉTAHEURISTIQUES D'APPRENTISSAGE, NOUVEAUX MODÈLES ET VISUALISATION DE DISSIMILARITÉ

Directeur de thèse : Professeur Mohamed SLIMANE

JURY :

M. Jin-Kao	HAO	Professeur	Université d'Angers
M. Jacques	LÉVY VÉHEL	Directeur de recherche	INRIA
M. Nicolas	MONMARCHÉ	Maître de conférences	Université François-Rabelais de Tours
M. Philippe	PREUX	Professeur	Université Lille 3
M. Patrick	SIARRY	Professeur	Université de Paris 12
M. Mohamed	SLIMANE	Professeur	Université François-Rabelais de Tours

UNIVERSITÉ FRANÇOIS-RABELAIS
TOURS

École Doctorale : Santé, Sciences
et Technologies

Année universitaire : 2004-2005

THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITÉ DE TOURS

Discipline : Informatique

présentée et soutenue publiquement

par :
Sébastien AUPETIT
le 30 novembre 2005

CONTRIBUTIONS AUX MODÈLES DE MARKOV CACHÉS : MÉTAHEURISTIQUES D'APPRENTISSAGE, NOUVEAUX MODÈLES ET VISUALISATION DE DISSIMILARITÉ

Directeur de thèse : Professeur Mohamed SLIMANE

JURY :

M. Jin-Kao	HAO	Président	Université d'Angers
M. Jacques	LÉVY VÉHEL	Éxaminateur	INRIA
M. Nicolas	MONMARCHÉ	Co-encadrant de thèse	Université François-Rabelais de Tours
M. Philippe	PREUX	Rapporteur	Université Lille 3
M. Patrick	SIARRY	Rapporteur	Université de Paris 12
M. Mohamed	SLIMANE	Directeur de thèse	Université François-Rabelais de Tours
M. Pierre	LIARDET	Invité	Université de Provence

Remerciements

« On ne voit bien qu'avec le cœur. L'essentiel est invisible pour les yeux.
– L'essentiel est invisible pour les yeux », répéta le petit prince, afin de se souvenir.

Antoine de Saint-Exupéry, Le Petit Prince

Ces remerciements sont l'occasion d'exprimer ce que je n'ai pas pu dire après la soutenance de thèse.

Au cours de l'année précédant le début de cette thèse, je suis allé discuter avec mon responsable de projet de fin d'études et de stage de DEA pour lui parler de la possibilité, pour moi, de m'engager dans une thèse. Je lui ai dit : « Je veux faire une thèse, mais seulement si c'est avec toi ! ». Il m'a alors informé qu'il n'avait pas de financement. Je lui ai dit : « C'est pas grave ! Je veux faire une thèse avec toi ! » et il a quand même accepté. Ce responsable, c'est Mohamed Slimane alias Mohand. Très vite, Mohand est devenu mon directeur de thèse et Nicolas Monmarché mon co-encadrant de thèse. J'aimerais donc les remercier tous les deux pour de nombreuses raisons : notamment, pour m'avoir pris en thèse avec eux, pour m'avoir fait confiance et pour m'avoir préparé au futur métier d'enseignant/chercheur. Mohand et Nicolas ne se sont pas confinés au rôle classique du directeur et du co-encadrant de thèse, ils ont fait plus : ils m'ont considéré plus comme un jeune collègue à conseiller que comme un thésard à diriger. Et pour cette noble attitude, je les remercie très sincèrement et j'espère être à la hauteur de leurs enseignements.

Je remercie les membres du jury devant lesquels j'ai soutenu cette thèse. En particulier, je salue le sérieux du travail et la pertinence des remarques émises par Philippe Preux et Patrick Siarry, dans leurs rôles de rapporteur. Je remercie également Jin-Kao Hao et Jacques Lévy Véhel pour leur participation à ce jury en tant qu'examinateurs. Finalement, je remercie tout particulièrement Pierre Liardet d'avoir accepté d'être l'invité de ce jury, même si sa qualité d'invité du jury n'est pas aussi bien reconnue que je le souhaite par les instances officielles. Je le remercie également pour son approche très (parfois trop ?) formelle et son enthousiasme dans nos collaborations mais aussi pour nos nombreux échanges stimulants qui cherchent à réconcilier l'informatique et les mathématiques.

Je remercie tous les enseignants, doctorants et étudiants du Département Informatique de Polytech'Tours pour m'avoir fait me sentir chez moi tout au long de ces années. Ces remerciements s'adressent plus particulièrement à Ali, Arnaud, Christophe, Nicolas, Mohand, Pierre, Sonia, Thierry, et les petits nouveaux, Alexis et Pierre, qui contribuent tous les jours à l'ambiance de l'équipe Handicap et Nouvelles Technologies dont je suis fier d'être membre. J'adresse également mes remerciements à Christian Proust pour sa bonne humeur, pour son amitié et pour m'avoir rendu fier d'être à Polytech'Tours et au Laboratoire d'Informatique.

D'un point de vue plus personnel, je remercie mes parents et ma soeur qui m'ont toujours soutenu dans les choix que j'ai effectué et qui ont quelques fois supportés mes absences et mes sautes d'humeurs. Pour terminer, je remercie Primice pour sa bonne humeur, pour son aide dans la correction du manuscrit et pour être venu me soutenir le jour J.

Rideau !

À ma famille et à tous ceux qui y ont cru.

Table des matières

Index des notations

Nous rappelons ici les différentes notations utilisées dans cette thèse :

$\forall$: quantificateur universel « quel que soit »

$\exists$: quantificateur existentiel « il existe »

$P(a/b)$: probabilité d'observer l'événement a conditionnellement à l'événement b

$\mathbb{X}$: dénomme un ensemble de valeurs

$\mathbb{R}$: ensemble des nombres réels

$O(N)$: la complexité est en N

$a_{i,j} = a_{ij}$: l'élément (i, j) de la matrice A

A' : la transposée de la matrice A

Λ : l'espace des MMC pour un nombre d'états cachés N fixé et un nombre de symboles M fixé

Introduction

Depuis les premiers travaux de A. A. Markov en 1913 sur les chaînes de Markov, presque 100 ans se sont écoulés. Initialement destinées à l'analyse du langage, les chaînes de Markov ont été utilisées pour de nombreuses applications. Cependant, elles ont rapidement montré leurs limites dans certains domaines. Les modèles de Markov cachés (MMC) ont alors été dérivé des chaînes de Markov afin de dépasser ces limites. Depuis lors, les MMC se sont vus dotés d'une théorie solide et d'un ensemble d'algorithmes permettant de résoudre, au moins partiellement, les problèmes associés. Cette théorie et ces algorithmes ont permis de faire des MMC un outil classique et performant d'intelligence artificielle. En tant que tels, les MMC participent à de nombreux systèmes d'intelligence artificielle (SIA) dans des domaines aussi vastes que le traitement d'images, la reconnaissance d'écriture, la biométrie, la biologie ou les nouvelles technologies de l'information et de la communication. Les MMC permettent, dans de nombreuses applications, d'obtenir de bons résultats, mais ils sont de plus en plus concurrencés par des outils tels que les réseaux de neurones ou les méthodes à noyaux. Pour contrer l'avancée des autres outils et rester compétitifs, les MMC doivent évoluer et être perfectionnés de manière à améliorer les performances des SIA.

Comment en est-on arrivé là ?

Cette thèse de doctorat a été menée sous la direction du professeur Mohamed Slimane et le co-encadrement du docteur Nicolas Monmarché, au sein de l'équipe Handicap et Nouvelles Technologies du Laboratoire d'Informatique (LI) de l'Université François-Rabelais de Tours. Cette thèse est le résultat de la convergence de plusieurs axes de recherche empruntés par M. Slimane et N. Monmarché depuis quelques années.

Sous l'encadrement de M. Slimane, en 1999 et 2000, deux doctorants, T. Brouard (Brouard, 1999) et N. Monmarché (Monmarché, 2000), terminaient leurs thèses. La première thèse abordait, entre autres, l'apprentissage de MMC à l'aide d'algorithmes génétiques tandis que la seconde établissait, entre autres, la métaheuristique de colonie de fourmis API et l'appliquait brièvement à l'apprentissage de MMC. Dans cette thèse, nous avons souhaité aller plus loin en étendant et en comparant ce panel de méthodes d'apprentissage (cf. chapitre 4). En 2002, je terminai un DEA (Aupetit, 2002) sous le co-encadrement de M. Slimane et N. Monmarché. Ce DEA consistait à exploiter des algorithmes de fourmis artificielles et des modèles de Markov cachés pour l'indexation de documents. Suite à une intuition partagée, nous avons conçu les modèles de Markov cachés à substitutions de symboles. Ces modèles ont montré depuis leur intéret pour la classification et la segmentation d'images (cf. chapitre 5). Depuis quelques années, des recherches sur la visualisation de données sont menées au sein du Laboratoire d'Informatique. M. Slimane et N. Monmarché font partie des chercheurs impliqués dans ces recherches. Nous avons donc naturellement cherché à appliquer les techniques de visualisation de données au cas très particulier des MMC (cf. chapitre 6). Finalement, notre implication, dès 2002, dans la mise en pratique d'environnements de parallélisation, et notre volonté de mettre à disposition d'autrui nos travaux, nous ont conduis à la création de la bibliothèque HMMTK (cf. chapitre 7). Ces différents aspects permettent d'améliorer les SIA.

Sujet de la thèse

Dans ce travail de thèse, nous nous sommes donc intéressés à l'amélioration des SIA utilisant des MMC. A cet effet, nous avons remarqué que l'amélioration des SIA peut être réalisée à trois niveaux différents : en amont, au coeur du système et en aval. En amont, une amélioration des pré-traitements sur les données peut être envisagée. Ces pré-traitements sont très dépendants des données manipulées et du domaine d'application du SIA. Au coeur du système, deux directions d'améliorations peuvent être considérées : l'amélioration de l'apprentissage et l'amélioration des modèles. Finalement, l'amélioration en aval peut consister en la meilleure compréhension de l'action des modèles sur les données afin d'améliorer les post-traitements. Cette thèse s'est concentrée sur trois directions différentes : l'amélioration de l'apprentissage des modèles, la création d'un nouveau modèle et la compréhension des modèles par la visualisation de dissimilarité.

La première direction que nous avons empruntée a pour objectif d'améliorer l'apprentissage des MMC. A cet effet, nous sommes partis du constat qu'il existe de nombreux algorithmes d'apprentissage de MMC spécialisés pour tel ou tel critère. Cependant, ces algorithmes ne permettent que d'améliorer un modèle initial de manière à mieux satisfaire le critère considéré. L'application répétée de l'un de ces algorithmes ne permet pas, dans le cas général, de trouver le meilleur modèle possible, mais uniquement un optimum local du critère. Un moyen de réduire ces effets consiste à utiliser des métaheuristiques pour s'efforcer de trouver le meilleur optimum local possible (*i.e.* l'optimum global). Afin de réduire le cadre de cette étude, nous nous sommes concentrés sur les métaheuristiques biomimétiques à base de population que sont les algorithmes génétiques, les algorithmes de fourmis artificielles et l'optimisation par essaim particulaire. Ces trois types de métaheuristiques ont été initialement définis pour des espaces de solutions de natures différentes. Les algorithmes génétiques s'appliquaient initialement à des espaces discrets tandis que l'optimisation par essaim particulaire s'appliquait initialement à des espaces continus. Les algorithmes de fourmis artificielles peuvent s'appliquer sur des espaces discrets et continus disposant de la notion de voisinage. Trois types d'espaces des solutions pour l'apprentissage de MMC peuvent alors être considérés. Le premier type d'espace est l'espace des triplets de matrices stochastiques qui est convexe, borné et à valeurs réelles. Le deuxième type d'espace correspond aux séquences d'états cachés, qui est discret et fini. Le troisième type d'espace est un espace continu, à valeurs réelles, convexe et muni d'une structure vectorielle. Les métaheuristiques biomimétiques que nous considérons dans cette thèse peuvent toutes être appliquées à ces trois types d'espaces de solutions. Certaines adaptations de ces métaheuristiques sont déjà présentes dans la littérature. Nous avons cherché dans cette thèse à évaluer quatre nouvelles adaptations sur ces espaces. Après avoir déterminé une bonne configuration des paramètres, nous avons comparé les performances des algorithmes.

La deuxième direction que nous avons empruntée consiste en la conception d'un nouveau modèle dérivé des MMC. De la même manière que les MMC ont été dérivés des chaînes de Markov, de très nombreux modèles dérivés des MMC sont apparus pour combler leurs limites. Parmi ceux-ci, on trouve les modèles de Markov doublement chaînés (Berchtold, 1999), les modèles de Markov multidimensionnels à processus indépendants (Brouard, 1999), les modèles de Markov hiérarchiques (Fine et al., 1998), les modèles de Markov cachés de profils (Eddy, 1998), les modèles de Markov cachés Input-Output (Bengio, 1999), les modèles de Markov cachés pseudo 2D (Agazzi and Kuo, 1993) et les arbres de Markov triplets (Pieczynski, 2003). Ces modèles ont un pouvoir d'expression supérieur aux MMC classiques, cependant la complexité des algorithmes associés est significativement supérieure à celle des algorithmes des MMC. Par conséquent, l'utilisation de ces nouveaux modèles ne s'effectue pas sans un coût calculatoire important. Il y a donc un besoin pour la création d'un modèle dérivé des MMC qui ait un pouvoir d'expression plus puissant, sans pour autant augmenter significativement les complexités des algorithmes. Pour cela, nous avons choisi de permettre l'inclusion de connais-

sances de l'expert dans les MMC. L'expert définit une matrice stochastique de probabilité de substitution de symboles. Le nouveau MMC, dénommé MMC à Substitutions de Symboles (MMCSS), possède autant de paramètres ajustables lors d'un apprentissage qu'un MMC ordinaire. La génération des symboles s'effectue quasiment de la même façon que pour les MMC, à l'exception que les symboles émis par le modèle (nommés méta-symboles) peuvent être remplacés par d'autres dans la séquence d'observations, selon une certaine loi de substitution. Cette possibilité de substitution permet, par exemple, de réduire les effets de manque d'information dans les séquences ou les effets dus à un ré-échelonnement des valeurs. Les algorithmes standards des MMC ont été adaptés à ces nouveaux modèles. Les algorithmes obtenus ont une complexité qui, du point de vue de la longueur de la séquence d'observations, est identique à celle des MMC. Nous avons également montré que les MMC ne sont qu'un cas particulier des MMCSS pour lesquels un méta-symbole est toujours substitué en lui-même. Finalement, nous avons mené les premières expérimentations sur l'apprentissage et la segmentation d'images qui ont montré que les MMCSS possèdaient des propriétés très prometteuses dans ce domaine.

La troisième direction que nous avons explorée consiste à essayer de mieux comprendre ce que modélisent réellement des MMC sur un jeu de données. A cet effet, nous avons considéré que les MMC pouvaient être mis en relation les uns avec les autres par l'intermédiaire d'une dissimilarité. Nous avons considéré une dissimilarité car une dissimilarité permet de manipuler les objets avec une très grande flexibilité. En effet, en choisissant bien la dissimilarité, il est possible d'étudier la présence, l'absence et/ou l'effet de telles ou telles caractéristiques sur les MMC. Nous nous sommes donc trouvés face au problème de la visualisation de dissimilarité avec un but explicatif. La visualisation de dissimilarité est une façon d'expliquer les influences. Expliquer les influences visuellement peut se faire de deux façons : faire émerger les groupes d'objets ou faire émerger les caractéristiques des objets. Dans le premier cas, de nombreuses méthodes existent, telles que la mise à l'échellonnement multidimensionnelle (*i.e. multidimensional scaling*) (Borg and Groenen, 1997) (Cox and Cox, 2001) ou la visualisation de graphe (Di Battista and Eades, 1994) (Herman et al., 2000). Dans le deuxième cas, il est nécessaire d'avoir recours à des techniques plus complexes utilisant les valeurs de la dissimilarité plutôt que les groupes qui en ressortent. Une solution classique utilisable en présence de données multidimensionelles est l'analyse factorielle simple et multiple, telle que l'analyse en composante principale (ACP). Dans le cas d'une dissimilarité, il n'est pas possible en général d'utiliser une ACP. Une solution possible est l'utilisation d'une ACP à noyau défini positif, mais il est rare, dans le cas d'une dissimilarité quelconque, de pouvoir se ramener à un noyau de ce type sans « déformation » de la dissimilarité. La solution que nous proposons est de généraliser l'ACP à noyau au cas des noyaux indéfinis. Cette ACP peut alors être utilisée dans une matrice de *scatterplots*[1] modifiée, afin d'analyser la dissimilarité, et par conséquent les MMC.

Organisation du document

Ce document est structuré en deux parties. La première partie établit plusieurs états de l'art sur les domaines abordés (chapitres 1, 2 et 3) tandis que la deuxième partie présente nos contributions aux modèles de Markov cachés (chapitres 4, 5, 6 et 7). Afin de faciliter sa lecture, voici un guide pour aborder les différents chapitres :

- le chapitre 1 introduit les modèles de Markov cachés, les algorithmes classiques ainsi que les principaux critères d'apprentissage de MMC ;
- le chapitre 2 établit un état de l'art des principales métaheuristiques existantes dans la littérature pour l'apprentissage de MMC ;

[1] Une matrice de *scatterplots* est une technique classique de visualisation disposant sous la forme d'une matrice, plusieurs graphes de nuages de points.

- le chapitre 3 dresse un panorama des principaux problèmes et techniques liés à la visualisation de données ;
- le chapitre 4 présente les différents espaces de solutions envisagés pour la conception de métaheuristiques pour l'apprentissage de MMC. Ces espaces sont utilisés pour concevoir quatre algorithmes d'apprentissage. Une recherche de bons paramètres est effectuée. Les algorithmes sont alors comparés avec ces configurations de bons paramètres ;
- le chapitre 5 définit les modèles de Markov cachés à substitutions de symboles (MMCSS) ainsi que les algorithmes associés. Deux expérimentations sur la classification et la segmentation d'images montrent finalement l'intérêt de ce nouveau type de MMC ;
- le chapitre 6 décrit la technique de l'analyse en composantes principales à noyaux indéfinis et son utilisation avec la matrice de *scatterplots* pseudo-euclidienne ;
- le chapitre 7 présente la bibliothèque HMMTK développée pendant la thèse pour la manipulation de MMC ;
- une conclusion sur l'ensemble de ces travaux ainsi que les perspectives d'avenir sont données à la fin du document.

Première partie

États de l'art

Chapitre 1

Modèles de Markov cachés

1.1 Introduction

Les modèles de Markov cachés sont des outils statistiques permettant de modéliser des phénomènes stochastiques. Ces modèles sont utilisés dans de nombreux domaines (Cappé, 2001) tels que la reconnaissance et la synthèse de la parole, la biologie, l'ordonnancement, l'indexation de documents, la reconnaissance d'images, la prédiction de séries temporelles, ... Pour pouvoir utiliser ces modèles efficacement, il est nécessaire d'en connaître les principes. Ce chapitre a donc pour objectif d'établir les principes, les notations utiles et les principaux algorithmes qui constituent la théorie des modèles de Markov cachés (MMC).

A cet effet, nous commençons ce chapitre en présentant un historique des étapes les plus marquantes dans la construction de cette théorie. Après avoir défini ce que sont les chaînes de Markov, nous montrons sur un exemple de lancers de pièces que ces dernières ne sont pas suffisantes pour bien modéliser ces lancers. Pour mieux modéliser ces lancers, il est nécessaire de considérer un modèle ayant un pouvoir d'expression supérieur. Les modèles de Markov cachés (MMC) en font partie. Nous présentons alors les MMC, ainsi que les notations associées que nous utilisons dans cette thèse. La suite du chapitre s'attache à présenter les algorithmes classiques des MMC : *Forward*, *Backward* et de *Viterbi*. La dernière section de ce chapitre est consacrée aux différents critères utilisables classiquement pour l'apprentissage de MMC. Finalement, nous terminons ce chapitre par plusieurs remarques sur les critères d'apprentissage, justifiant les travaux présentés aux chapitres 2 et 4.

1.2 Historique

Les modèles de Markov cachés ont une longue histoire derrière eux. En 1913, les premiers travaux sur les chaînes de Markov pour l'analyse du langage permettent à A. A. Markov de concevoir la théorie des chaînes de Markov (Markov, 1913). De 1948 à 1951, Shannon conçoit la théorie de l'information en utilisant les chaînes de Markov (Shannon, 1948) (Shannon, 1951). Dès 1958, les modèles probabilistes d'urnes (Feller, 1958), le calcul direct du maximum de vraisemblance (Hartley, 1958) et l'observation de la suite d'états dans une chaîne de Markov (Billingsley, 1961) sont réalisés. Mais ce n'est qu'à partir de 1966 avec les travaux de L. E. Baum (Baum and Petrie, 1966) (Baum and Eagon, 1967) (Baum and Sell, 1968) (Petrie, 1969) (Baum et al., 1970) (Baum, 1972) que les algorithmes basiques pour l'estimation des états et des paramètres des modèles, pour les modèles de Markov cachés, sont mis au point. À partir de 1980, ces modèles sont étendus afin d'intégrer la notion de durée variable (Furguson, 1980) et des densités de probabilités continues multivariables (Liporace, 1982). Les travaux de A. J. Viterbi (Viterbi, 1967) et G. D. Forney (Forney Jr., 1973) ont permis de construire un algorithme efficace et dont la complexité est linéaire, par rapport à la longueur de la suite

d'observations, pour le calcul de la séquence d'états cachés. En 1970, les termes « modèles de Markov cachés » ou « chaînes de Markov cachées » (*hidden Markov models* en anglais) sont inventés par L. P. Neuwirt afin de remplacer l'appellation « fonction probabiliste d'une chaîne de Markov » utilisée jusque là (Slimane, 2002).

À partir de 1975, les modèles de Markov cachés ont commencé à être utilisés dans de nombreux domaines, comme par exemple la reconnaissance automatique de la parole (Rabiner, 1989) (Kriouile, 1990). Les premiers travaux sur les modèles de Markov cachés pour la reconnaissance automatique de la parole ont été menés en parallèle par le groupe IBM composé de L. R. Bahl et F. Jelinek (Bahl and Jelinek, 1975) (Jelinek et al., 1975) et par J. K. Baker au CMU (Baker, 1975a) (Baker, 1975b). Ces travaux ont permis de découvrir les capacités des modèles de Markov cachés pour la reconnaissance de la parole.

Dans les années 1980, les modèles de Markov cachés incorporant des réseaux de neurones apparaissent (Bourland and Wellekens, 1990). Depuis lors, ces nouveaux modèles ont été très largement utilisés pour la reconnaissance de mots isolés (Rabiner et al., 1983) (Juang and Rabiner, 1986) (Poritz and Richter, 1986) (Euler and Wolf, 1988), pour la reconnaissance de mots enchaînés (Rabiner and Levinson, 1985), pour la reconnaissance de la parole continue (Bahl et al., 1983) ou pour la localisation de mots dans une phrase (Rosemberg and Colla, 1987) (Dours, 1989).

À partir des années 1990, les premières applications à la reconnaissance d'images (Samaria and Harter, 1994) et de l'écriture apparaissent (Kundu and Bahl, 1988) (Olivier et al., 1993) (El-Yacoubi et al., 1999).

Récemment, les modèles de Markov cachés ont même été utilisés pour l'ordonnancement de tâches (Soukhal et al., 2001a) (Soukhal et al., 2001b) et les technologies de l'information (Zaragoza and Gallinari, 1998) (Amini, 2001) (Serradura et al., 2001).

Comme nous l'avons évoqué précédemment, nous ne nous attarderons pas sur les applications des modèles de Markov cachés, cependant le lecteur intéressé par une liste bibliographique sur les utilisations de ces modèles, des années 1980 jusqu'aux années 2001, pourra se référer à (Cappé, 2001), qui référence plusieurs travaux en acoustique, en biologie, en climatologie, en économétrie, en reconnaissance de l'écriture et de la parole et en traitement d'images et de signaux. Des travaux sur les systèmes de contrôle, sur les systèmes de communication, sur la vision par ordinateur et sur le traitement du signal et de la parole sont également référencés.

1.3 Définitions

Les modèles de Markov cachés sont une famille d'outils mathématiques probabilistes parfaitement adaptés à la modélisation de séquences temporelles. Il existe plusieurs types de modèles de Markov cachés afin de mieux répondre à des problèmes spécifiques. Dans le cadre de cette étude et plus particulièrement de ce chapitre, nous nous intéresserons principalement aux modèles de Markov cachés discrets du premier ordre, que nous abrégerons par la suite en MMC. Pour pouvoir présenter les MMC, il est nécessaire de commencer par présenter les modèles de Markov et les propriétés qui leur sont associées.

1.3.1 Les chaînes de Markov discrètes

En calcul des probabilités, on définit une variable aléatoire (v.a.) réelle comme une fonction mesurable $X : \Omega \rightarrow \mathbb{R}$. Ω est appelé l'univers. Dans de nombreux cas de figures, Ω est l'ensemble des réels $\mathbb{R}$, l'ensemble des entiers positifs $\mathbb{N}$ ou un de leurs sous-ensembles.

Définition 1 *Un processus stochastique est une famille* $\{X_t\}_{t\in\mathbb{T}}$ *de v.a. définies sur* Ω *(*$X_t : \Omega \rightarrow \mathbb{R}$*).*

L'ensemble $\mathbb{T}$ représente souvent la notion de temps mais il peut également correspondre à la notion de position spatiale en dimension 2 ou à toute autre notion en autant de dimensions que nécessaire. Dans le cas où $\mathbb{T}$ représente la notion de temps et si $\mathbb{T}$ est discret, on parle de processus stochastique en temps discret, tandis que le processus est dit en temps continu, lorsque $\mathbb{T}$ est continu.

Définition 2 *Les états d'un processus stochastique défini par les v. a. $X_t : \Omega \to \mathbb{R}$ pour tout $t \in \mathbb{T}$ sont les valeurs prises par ces v.a. lorsque t varie. On note $\mathbb{S}$ l'ensemble des « états » du processus.*

A. A. Markov fut le premier à étudier et à poser les bases mathématiques permettant l'étude des chaînes qui portent son nom. La définition de ces chaînes est la suivante :

Définition 3 *Un processus $\{S_t\}_{t\in\mathbb{T}}$ ($S_t : \Omega \to \mathbb{S}$) est une chaîne de Markov s'il vérifie les trois conditions suivantes :*

1. *$\mathbb{T}$ est dénombrable ou fini. Dans ce cas et pour simplifier les notations ultérieures, il est toujours possible de prendre $\mathbb{T} \subseteq \mathbb{N}^* = \{1, 2, \ldots\}$. Cette condition signifie que le processus ne change de valeur qu'à des instants déterminés a priori.*
2. *L'ensemble $\mathbb{S}$ des états du processus est dénombrable. Dans la suite, nous supposerons également que $\mathbb{S}$ est fini. Nous pouvons alors définir $\mathbb{S} = \{s_1, \ldots, s_N\}$ cet ensemble.*
3. *Le processus est associé à une fonction de probabilité P vérifiant la propriété markovienne : « la probabilité que le processus soit dans un état particulier à un instant t ne dépend que de l'état dans lequel se trouve le processus au temps $t-1$ ». Soit $Q = (q_t)_{t\in\mathbb{T}}$ une suite d'états du processus ($q_t \in \mathbb{S}$). La propriété de Markov vérifie la relation suivante, pour toute suite d'états Q et pour tout instant $t \in \mathbb{T}$:*

$$P(S_t = q_t / S_{t-1} = q_{t-1}, \ldots, S_1 = q_1) = P(S_t = q_t / S_{t-1} = q_{t-1})$$

La probabilité $P(S_t = q_t / S_{t-1} = q_{t-1})$ correspond à la probabilité de transition de l'état q_{t-1} à l'instant $t-1$ vers l'état q_t à l'instant t.

Définition 4 *Une chaîne de Markov est homogène (dans le temps) si et seulement si les probabilités de transition ne dépendent pas du temps t (les probabilités de transition sont stationnaires), c'est-à-dire que pour tout $(t, t') \in \mathbb{T}^2$, on a :*

$$P(S_{t+1} = s_j / S_t = s_i) = P(S_{t'+1} = s_j / S_{t'} = s_i)$$

On note $a_{i,j}$ cette probabilité.

Une chaîne de Markov homogène est donc totalement définie par la donnée des états, des probabilités des états initiaux Π et des probabilités des transitions entre états A avec :

- $\Pi = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_N \end{pmatrix} = (\pi_1, \ldots, \pi_N)'$ et $\pi_i = P(S_1 = s_i)$
- $A = (a_{i,j})_{1\leq i,j\leq N}$ et $a_{i,j} = P(S_{t+1} = s_i / S_t = s_j)$

Définition 5 *Vecteurs et matrices stochastiques :*

- *Un vecteur $V = (v_1, \ldots, v_N)$ de dimension N (ou, de manière équivalente, son transposé) est stochastique si et seulement si :*
  - *pour tout $i = 1..N$, $0 \leq v_i \leq 1$,*
  - *$\sum_{i=1}^{N} v_i = 1$.*

- *Une matrice* $M = (m_{i,j})_{1 \leq i,j \leq N}$ *de dimension* $N \times N$ *est dite stochastique si et seulement si :*
  - *pour tout* $i = 1..N$ *et* $j = 1..N$, $0 \leq m_{i,j} \leq 1$,
  - *pour tout* $i = 1..N$, $\sum_{j=1}^{N} m_{i,j} = 1$.

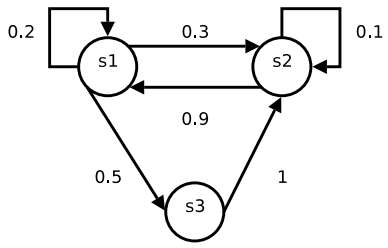
Caractéristiques d'une chaîne de Markov :
- Une matrice est stochastique si et seulement si les lignes qui la composent sont des vecteurs stochastiques.
- Le système est forcément dans un et un seul état particulier au départ donc Π est un vecteur stochastique. A est une matrice stochastique car, en partant d'un état s_i à l'instant t, le processus transite forcément vers l'un des N états du système au temps $t+1$.
- A tout couple formé d'un vecteur stochastique V de dimension N et d'une matrice stochastique M de dimension $N \times N$, il est possible d'associer une chaîne de Markov caractérisée par le couple (V, M).

Définition 6 *Une chaîne de Markov peut être représentée graphiquement. Pour cela, on associe à la chaîne de Markov* $\{S_t\}_{t \in \mathbb{T}}$ *un graphe* G *dont l'ensemble des sommets est en bijection avec l'ensemble des états* $\mathbb{S}$ *et dont l'ensemble des arcs* U *(orientés dans le sens des transitions) est défini par*

$$(s_i, s_j) \in U \Leftrightarrow a_{i,j} > 0$$

Afin de simplifier les notations, l'ensemble des sommets du graphe G *sera représenté par l'ensemble* $\mathbb{S}$.

La figure 1.1 présente la représentation graphique associée à la chaîne de Markov (Π, A).



$$A = \begin{pmatrix} 0.2 & 0.3 & 0.5 \\ 0.9 & 0.1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \qquad \Pi = \begin{pmatrix} 0.4 \\ 0.2 \\ 0.4 \end{pmatrix}$$

FIG. 1.1 – Représentation graphique de la chaîne de Markov (Π, A).

Les chaînes de Markov peuvent servir à modéliser de nombreux processus stochastiques. Voici quelques exemples :
- modélisation des jours de pluie et de soleil sous certains climats. Pour cela, on suppose qu'une journée ne peut être que pluvieuse ou qu'ensoleillée et on estime avec une distribution de Bernouilli les probabilités en fonction du jour t : soleil avec une probabilité $p(t)$ et pluie avec une probabilité $1 - p(t)$ avec l'hypothèse que le temps au jour t ne dépend que du temps au jour $t-1$ et que, sur une période assez longue, il y a une certaine stabilité statistique du temps (homogénéité du processus) ;
- modélisation du nombre d'absents et de présents dans une entreprise (en dehors des épidémies et des périodes de vacance). Ce nombre change aléatoirement chaque jour. On a $S_t = S_{t-1} + \Delta S_t$. On suppose alors que ΔS_t ne dépend que de S_{t-1} ;

- modélisation de file d'attente : S_t correspond au nombre de personnes dans la file d'attente à la fin de la t-ième période de temps ;
- modélisation de l'état d'usure S_t d'un équipement à la période t ;
- modélisation stochastique de la génération de la parole utilisée en reconnaissance de la parole par ordinateur. Par exemple, S_t correspond à la classification grammaticale (substantif, adjectif, verbe, pronom, ...) du t-ième mot de la phrase ;
- modélisation et traitement d'images numériques, S_t, correspondant à « l'état » d'un pixel (son niveau de gris par exemple), est supposé ne dépendre que des quelques pixels « précédents » (le sens de « précédent » étant ici associé au sens du balayage de l'image ou à la définition d'un voisinage) ;

Cependant, dans certains cas, ces modèles ne permettent pas d'exprimer le comportement du système avec suffisamment de précision. Pour améliorer cette précision, les modèles de Markov cachés ont été développés.

1.3.2 Les modèles de Markov cachés discrets (MMC)

Pour introduire les modèles de Markov cachés discrets, le moyen le plus simple est de partir d'un exemple.

1.3.2.1 Un exemple : Pierre et Paul jouent avec des pièces

Pierre et Paul veulent jouer à un jeu avec des pièces de monnaie. Pierre et Paul se trouvent de chaque côté d'un mur. Aucun d'entre eux ne voit ce que fait l'autre. Pierre possède 3 pièces biaisées. Pierre choisit une première pièce, la lance et énonce à Paul le côté visible après que la pièce soit tombée sur le sol. Paul entend « Face » mais il ne sait pas quelle pièce a été choisie par Pierre. Pierre remet les trois pièces ensemble et en choisit une nouvelle qu'il lance. Il énonce le résultat et recommence plusieurs fois. Pierre n'aime pas beaucoup le hasard, il a décidé qu'il choisirait les pièces avec une certaine loi de probabilité. Pour cela, il s'est fixé la probabilité de choisir une pièce en fonction de son choix précédent et uniquement de celui-ci. De l'autre côté du mur, Paul ne sait que deux choses sur ce que fait Pierre : il sait qu'il joue avec trois pièces et que les résultats des lancers sont « FFPPPPPFPFPFFPPPPF » avec F pour « Face » et P pour « Pile ». La figure 1.2 présente la situation. L'objectif de Paul est alors de modéliser et prédire ce que va faire Pierre.

Comme on peut le remarquer dans un premier temps, ce jeu peut être modélisé par un modèle de Markov. En effet, les états du système associé pourraient correspondre aux couples (pièce, résultat du lancer). Cependant, l'absence de connaissances sur la séquence des pièces rend difficile l'estimation des probabilités du modèle. De plus, même avec une bonne estimation des probabilités, le modèle obtenu n'est pas très proche de la réalité car les pièces sont biaisées. Si les pièces sont biaisées de la même façon, le choix des pièces par Pierre n'a aucune influence sur les résultats du jeu. Dans ce cas, le système peut se réduire au lancer d'une pièce dont le biais est statistiquement estimable et par conséquent une chaîne de Markov serait suffisante pour modéliser le système.

Admettons que Paul ait réussi à trouver un bon modèle du jeu de Pierre. Pierre donne alors une séquence de « Pile » et « Face » à Paul. Comme Pierre est très joueur, Paul se doute que la séquence peut provenir d'un nouvel ensemble de lancers avec les mêmes pièces et les mêmes lois de choix, d'un autre jeu avec d'autres pièces et lois ou de l'imagination de Pierre. La tâche de Paul est alors de déterminer si cette séquence est issue des mêmes pièces et lois qu'au premier jeu.

Comme nous le verrons plus loin, les problèmes que Paul a à résoudre sont difficiles mais ils peuvent néanmoins être résolus sous certaines conditions. A cet effet, il est nécessaire d'introduire une extension des modèles de Markov : les modèles de Markov cachés.

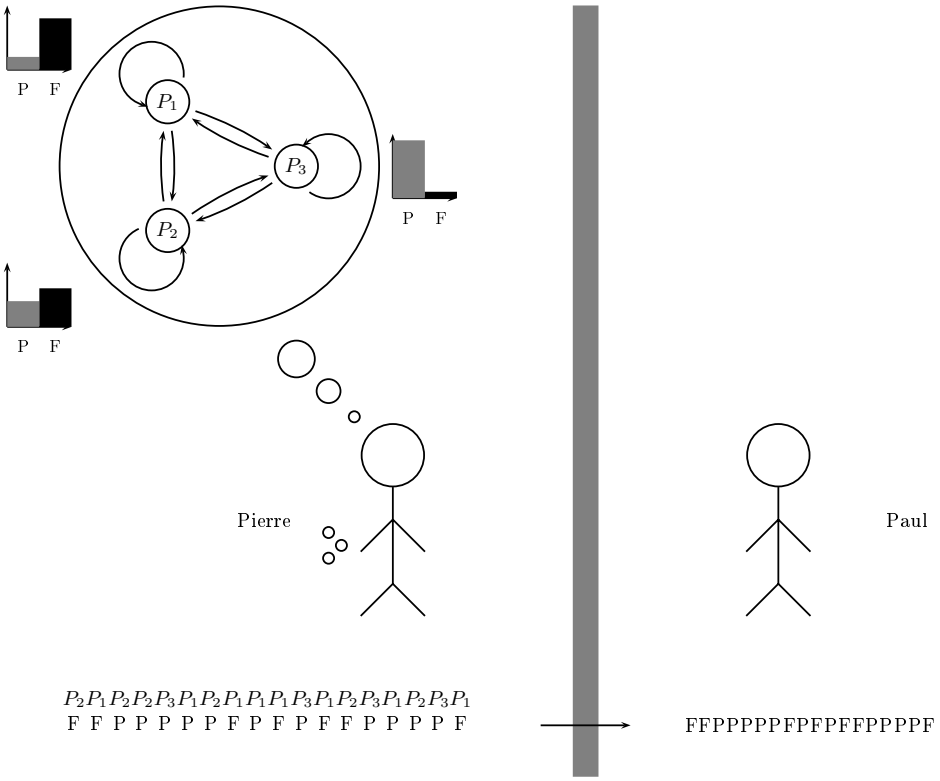


FIG. 1.2 – Pierre et Paul jouent avec des pièces.

1.3.2.2 Définitions

Un modèle de Markov caché discret correspond à la modélisation de deux processus stochastiques : un processus caché parfaitement modélisé par une chaîne de Markov discrète et un processus observé dépendant des états du processus caché.

Définition 7 *Soit* $\mathbb{S} = \{s_1, \ldots, s_N\}$ *l'ensemble des* N *états cachés du système. Soit* $S = (S_1, \ldots, S_T)$ *un* T*-uple de v.a. définies sur* $\mathbb{S}$*. Soit* $\mathbb{V} = \{v_1, \ldots, v_M\}$ *l'ensemble des* M *symboles émissibles par le système. Soit* $V = (V_1, \ldots, V_T)$ *un* T*-uple de v.a. définies sur* $\mathbb{V}$*.*

Un modèle de Markov caché discret du premier ordre est alors défini par les probabilités suivantes :

- *les probabilités d'initialisation des états cachés :* $P(S_1 = s_i)$
- *les probabilités de transition entre états cachés :* $P(S_t = s_j / S_{t-1} = s_i)$
- *les probabilités d'émission des symboles dans chaque état caché :* $P(V_t = v_j / S_t = s_i)$.

Si le modèle de Markov caché est stationnaire alors les probabilités de transition entre états cachés et les probabilités d'émission des symboles dans chaque état caché sont indépendantes du temps $t > 1$. On peut alors définir, pour tout $t > 1$ quelconque, $A = (a_{i,j})_{1 \leq i,j \leq N}$ avec $a_{i,j} = P(S_t = s_j / S_{t-1} = s_i)$, $B = (b_i(j))_{1 \leq i \leq N, 1 \leq j \leq M}$ avec $b_i(j) = P(V_t = v_j / S_t = s_i)$ et $\Pi = (\pi_1, \ldots, \pi_N)'$ avec $\pi_i = P(S_1 = s_i)$. Un modèle de Markov caché stationnaire du premier ordre λ est donc totalement défini par le triplet (A, B, Π). Par la suite, nous utiliserons la notation $\lambda = (A, B, \Pi)$ et nous emploierons le terme MMC pour des modèles de Markov cachés stationnaires du premier ordre. Les relations de dépendance entre les différentes variables aléatoires d'un MMC sont schématisées par la figure 1.3. Dans cette représentation, les flèches partent de la v.a. qui conditionne et se terminent au niveau de la variable aléatoire conditionnée. Dans la figure, seules les transitions au temps $t-1$, t et $t+1$ sont représentées.

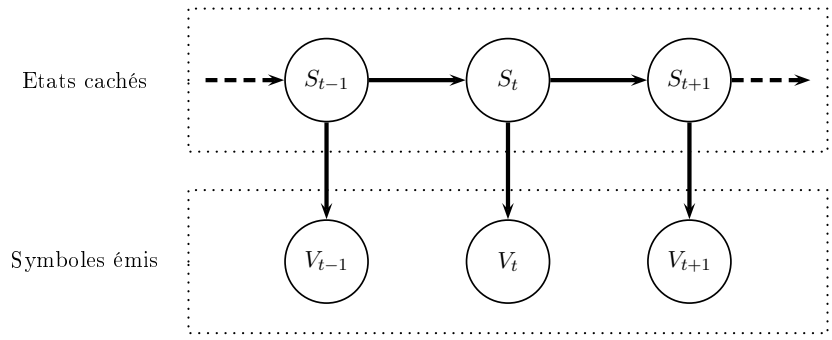


FIG. 1.3 – Relations de dépendance entre les variables aléatoires d'un MMC.

On note $Q = (q_1, \ldots, q_T) \in \mathbb{S}^T$ une séquence d'états cachés et $O = (o_1, \ldots, o_T) \in \mathbb{V}^T$ une séquence de symboles observés. La probabilité de réalisation de la séquence d'états cachés Q et de la séquence d'observations O par rapport au MMC λ est alors

$$P(V = O, S = Q/A, B, \Pi)$$

ou plus simplement[1]

$$P(V = O, S = Q/\lambda)$$

En utilisant les dépendances des probabilités conditionnelles, on déduit que :

$$P(V = O, S = Q/\lambda) = P(V = O/S = Q, \lambda)P(S = Q/\lambda)$$

De plus,

$$P(V = O/S = Q, \lambda) = \prod_{t=1}^{T} P(V_t = o_t/S_t = q_t, \lambda)$$

$$P(S = Q/\lambda) = P(S_1 = q_1/\lambda) \prod_{t=1}^{T-1} P(S_{t+1} = q_{t+1}/S_t = q_t, \lambda)$$

A partir d'un MMC λ, d'une séquence d'états cachés Q et d'une séquence d'observations O, il est possible de calculer l'adéquation entre le modèle λ et les deux séquences Q et O. Pour cela, il suffit de calculer la probabilité $P(V = O, S = Q/\lambda)$. Cette dernière correspond à la probabilité que la séquence d'observations O ait effectivement été engendrée par le modèle λ en suivant la séquence d'états cachés Q.

Lorsque la séquence d'états cachés n'est pas connue, il est possible d'évaluer la vraisemblance d'une séquence d'observations O par rapport à un modèle λ. La vraisemblance correspond à la probabilité $P(V = O/\lambda)$ que la séquence d'observations ait été engendrée par le modèle pour l'ensemble des séquences d'états cachés possibles. On remarque alors que la formule suivante est vérifiée :

$$P(V = O/\lambda) = \sum_{Q \in \mathbb{S}^T} P(V = O, S = Q/\lambda)$$

Les MMC, pour être utiles, nécessitent la résolution de plusieurs problèmes principaux : le calcul de la vraisemblance, le décodage/segmentation de séquence d'observations et l'apprentissage.

[1] Formellement, il faudrait considérer λ comme la réalisation d'une v.a. $P(V = O, S = Q/l = \lambda)$. Mais, afin de simplifier les notations, nous avons choisi d'omettre la variable aléatoire exprimant le modèle.

1.4 La calcul de la vraisemblance

Comme nous l'avons vu précédemment, calculer la vraisemblance d'une séquence de T observations par rapport à un MMC λ consiste à évaluer la probabilité $P(V = O/\lambda)$. Ce calcul peut s'effectuer de plusieurs façons : naïvement, par un calcul direct, avec une complexité en $O(TN^T)$ ou efficacement, grâce à l'algorithme *Forward* (Rabiner, 1989).

1.4.1 L'algorithme *Forward*

Pour présenter rapidement cet algorithme, il est nécessaire de définir les variables *Forward* (pour tout $i = 1..N$ et $t = 2..T$) :

$$\begin{cases} \alpha_1(i) = P(V_1 = o_1, S_1 = s_i/\lambda) \\ \alpha_t(i) = P(V_1 = o_1, \ldots, V_t = o_t, S_t = s_i/\lambda) \end{cases}$$

On remarque alors que la relation de récurrence suivante est vérifiée pour tout $t = 1..T-1$ et $j = 1..N$.

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij}$$

De plus, on a $P(V = O/\lambda) = \sum_{i=1}^{N} \alpha_T(i)$. L'algorithme *Forward* est alors donné par l'algorithme 1.1. La complexité de cet algorithme est en $O(N^2T)$.

Algorithme 1.1: Algorithme *Forward*

Pour $i = 1$ **à** N **Faire**
  $\alpha_1(i) = \pi_i b_i(o_1)$
Fin Pour
Pour $t = 1$ **à** $T-1$ **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(o_{t+1})$
  Fin Pour
Fin Pour
$P(V = O/\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Cet algorithme permet de calculer la vraisemblance d'une séquence d'observations. Cependant, dans la pratique, des problèmes de précision numérique apparaissent à l'implémentation rendant l'algorithme *Forward* inutilisable. Une solution[2] consiste à opérer un ré-échelonnement des valeurs (Rabiner, 1989). Pour cela, on définit deux ensembles de variables $\alpha'_t(i)$ et $\tilde{\alpha}_t(i)$ (pour tout $i = 1..N$ et $t = 2..T$) par :

$$\begin{cases} \alpha'_1(i) = \alpha_1(i) \\ \alpha'_t(i) = \sum_{j=1}^{N} \tilde{\alpha}_{t-1}(j) a_{ij} b_j(o_t) \end{cases}$$

On définit c_t (pour tout $t = 1..T$) le coefficient de normalisation à 1 de la somme des $\tilde{\alpha}_t(i)$ (*i.e.* $\sum_{i=1}^{N} \tilde{\alpha}_t(i) = 1$) par $c_t = \frac{1}{\sum_{i=1}^{N} \alpha'_t(i)}$. On pose $\tilde{\alpha}_t(i) = c_t \alpha'_t(i)$. Avec $C_t = \prod_{k=1}^{t} c_k$, on montre par récursivité que

$$\tilde{\alpha}_t(i) = C_t \alpha_t(i)$$

[2]Dans (Kriouile, 1990), il est proposé une autre méthode effectuant la normalisation non pas sur la somme, mais sur le maximum.

Or, par définition, on a $\sum_{i=1}^{N} \tilde{\alpha}_t(i) = 1$ d'où :

$$P(V = O/\lambda) = \frac{1}{C_T}$$

L'algorithme *Forward* avec ré-échelonnement (également nommé *rescaling*) (Rabiner, 1989) est donné par l'algorithme 1.2. Sa complexité est identique à celle de l'algorithme *Forward*, c'est-à-dire $O(N^2T)$, cependant l'algorithme nécessite plus d'opérations. De plus, la valeur prise par $P(V = O/\lambda)$ est très petite et est considérée la plupart du temps comme étant nulle dans les représentations en nombres réels sur les machines. Par conséquent, on considère plus facilement son logarithme, qui s'obtient par

$$\begin{aligned} \ln P(V = O/\lambda) &= \ln \frac{1}{C_T} \\ &= -\sum_{t=1}^{T} \ln c_t \end{aligned}$$

Algorithme 1.2: Algorithme *Forward* avec ré-échelonnement

Pour $i = 1$ **à** N **Faire**
  $\alpha'_1(i) = \pi_i b_i(o_1)$
Fin Pour
$c_1 = \frac{1}{\sum_{i=1}^{N} \alpha'_1(i)}$
Pour $i = 1$ **à** N **Faire**
  $\tilde{\alpha}_1(i) = c_1 \alpha'_1(i)$
Fin Pour
Pour $t = 1$ **à** $T - 1$ **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\alpha'_{t+1}(j) = \left(\sum_{i=1}^{N} \tilde{\alpha}_t(i) a_{ij}\right) b_j(o_{t+1})$
  Fin Pour
  $c_t = \frac{1}{\sum_{j=1}^{N} \alpha'_t(j)}$
  Pour $j = 1$ **à** N **Faire**
    $\tilde{\alpha}_{t+1}(j) = c_t \alpha'_{t+1}(j)$
  Fin Pour
Fin Pour
$P(V = O/\lambda) = \frac{1}{\prod_{t=1}^{T} c_t}$

1.4.2 L'algorithme *Backward*

Bien que le problème du calcul de la vraisemblance soit résolu, nous allons également présenter l'algorithme *Backward* (Rabiner, 1989) qui permet aussi de calculer la vraisemblance et qui surtout sera nécessaire dans les sections et chapitres ultérieurs, notamment pour l'apprentissage. Les variables *Backward* sont définies par (pour tout $i = 1..N$ et $t = 1..T-1$) :

$$\begin{cases} \beta_T(i) = 1 \\ \beta_t(i) = P(V_{t+1} = o_{t+1}, \ldots, V_T = o_T / S_t = s_i / \lambda) \end{cases}$$

Pour tout $i = 1..N$ et $t = 1..T-1$, les relations suivantes sont vérifiées :

$$\begin{aligned} \beta_t(i) &= \sum_{i=1}^{N} a_{ij}\beta_{t+1}(i)b_i(o_{t+1}) \\ P(V = O/\lambda) &= \sum_{i=1}^{N} \pi_i b_i(o_1)\beta_1(i) \end{aligned}$$

L'algorithme *Backward*, de même complexité que l'algorithme *Forward*, est donné par l'algorithme 1.3.

Algorithme 1.3: Algorithme *Backward*

Pour $i = 1$ **à** N **Faire**
  $\beta_T(i) = 1$
Fin Pour
Pour $t = T-1$ **à** 1 **Faire**
  Pour $i = 1$ **à** N **Faire**
    $\beta_t(i) = \sum_{j=1}^{N} a_{ij}\beta_{t+1}(j)b_j(o_{t+1})$
  Fin Pour
Fin Pour
$P(V = O/\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1)\beta_1(i)$

Tout comme l'algorithme *Forward*, l'algorithme *Backward* souffre de problème de précision numérique. Par conséquent, il est nécessaire d'utiliser le ré-échelonnement des variables *Backward*. Pour cela, on définit l'ensemble de variables $\tilde{\beta}_t(i)$ par (pour tout $i = 1..N$ et $t = 1..T-1$) :

$$\begin{cases} \tilde{\beta}_T(i) = c_T \\ \tilde{\beta}_t(i) = c_t \sum_{j=1}^{N} a_{ij} b_j(o_{t+1})\tilde{\beta}_{t+1}(j) \end{cases}$$

On pourra remarquer que les coefficients c_t sont ceux calculés précédemment pour l'algorithme *Forward* avec ré-échelonnement.

En définissant $D_t = \prod_{k=t}^{T} c_k$, il est possible de montrer les relations suivantes :

$$\begin{aligned} D_1 &= C_T \\ &= \frac{1}{P(V = O/\lambda)} \\ C_t D_{t+1} &= C_T \\ &= \frac{1}{P(V = O/\lambda)} \\ \tilde{\beta}_t(i) &= D_t \beta_t(i) \end{aligned}$$

L'algorithme *Backward* avec ré-échelonnement (Rabiner, 1989) est donné par l'algorithme 1.4. Sa complexité est identique à celle de l'algorithme *Backward*, c'est-à-dire $O(N^2T)$. On remarque également que le calcul de $P(V = O/\lambda)$ par cet algorithme offre peu d'intérêt, car il nécessite de connaître les coefficients c_t de l'algorithme *Forward* avec ré-échelonnement.

1.4.3 Probabilités déductibles

A partir des variables *Forward* et *Backward*, avec ou sans ré-échelonnement, il nous est d'ores et déjà possible d'exprimer deux probabilités utiles dans les sections et chapitres suivants.

$$\begin{aligned} P(V = O, S_t = s_i/\lambda) &= \alpha_t(i)\beta_t(i) \\ &= \frac{\tilde{\alpha}_t(i)\tilde{\beta}_t(i)}{P(V = O/\lambda)} \end{aligned} \tag{1.1}$$

$$\begin{aligned} P(V = O, S_t = s_i, S_{t+1} = s_j/\lambda) &= \alpha_t(i)a_{i,j}b_j(o_{t+1})\beta_{t+1}(j) \\ &= \frac{\tilde{\alpha}_t(i)a_{i,j}b_j(o_{t+1})\tilde{\beta}_{t+1}(j)}{P(V = O/\lambda)} \end{aligned} \tag{1.2}$$

Algorithme 1.4: Algorithme *Backward* avec ré-échelonnement

Pour $i = 1$ **à** N **Faire**
  $\tilde{\beta}_T(i) = c_T$
Fin Pour
Pour $t = T - 1$ **à** 1 **Faire**
  Pour $i = 1$ **à** N **Faire**
    $\tilde{\beta}_t(i) = c_t \sum_{j=1}^{N} a_{i,j}\tilde{\beta}_{t+1}(j)b_j(o_{t+1})$
  Fin Pour
Fin Pour

1.5 Décodage/segmentation de séquences d'observations

Le décodage ou la segmentation de séquences d'observations consiste à trouver la séquence d'états cachés qui a engendré une séquence d'observations. Deux approches sont possibles. La première consiste à rechercher, à chaque instant, l'état qui a le plus probablement engendré le symbole observé. La deuxième approche consiste à trouver la séquence complète d'états cachés qui a le plus probablement engendré la séquence d'observations.

1.5.1 États cachés les plus probables à chaque instant

Dans cette approche, on cherche la séquence $Q^* = (q_1^*, \ldots, q_T^*) \in \mathbb{S}^T$ vérifiant, pour tout $t = 1..T$, l'équation :

$$q_t^* = \arg\max_{i=1..N} P(V = O, S_t = s_i/\lambda)$$

Il est donc nécessaire, d'après la formule 1.1, de calculer en premier lieu les variables *Forward* et *Backward*. Malgré sa formulation simple, la recherche de l'état caché le plus probable à chaque instant a une complexité en $O(N^2T)$. De plus, la séquence Q^* obtenue peut être inconsistante, dans le sens où $P(V = O, S = Q^*) = 0$. En effet, il est possible que la transition entre deux états s_i et s_j existe dans la séquence Q^*, alors que la probabilité $a_{i,j}$ est nulle.

1.5.2 Algorithme de Viterbi

La recherche de la séquence d'états cachés Q^* qui a le plus probablement engendré une séquence d'observations O consiste à résoudre

$$Q^* = \arg\max_{Q \in \mathbb{S}^T} P(V = O, S = Q/\lambda)$$

L'algorithme permettant de résoudre ce problème est l'algorithme de Viterbi (Viterbi, 1967). On définit

$$\delta_t(i) = \max_{(q_1,\ldots,q_{t-1}) \in \mathbb{S}^{t-1}} \{P(S_1 = q_1, \ldots, S_{t-1} = q_{t-1}, S_t = s_i, V_1 = o_1, \ldots, V_t = o_t/\lambda)\}$$

la probabilité du meilleur chemin partiel amenant à l'état caché s_i au temps t et $\psi_t(i)$ le meilleur chemin amenant à l'état s_i au temps t à partir du temps $t-1$. L'algorithme de Viterbi est alors donné par l'algorithme 1.5. Sa complexité est $O(N^2T)$.

Algorithme 1.5: Algorithme de Viterbi

Pour $i = 1$ **à** N **Faire**
  $\delta_1(i) = \pi_i b_i(o_1)$
Fin Pour
Pour $t = 2$ **à** T **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\psi_t(j) = \arg \max_{1 \leq i \leq N} \{\delta_{t-1}(i) a_{ij}\}$
    $\delta_t(j) = \max_{1 \leq i \leq N} \{\delta_{t-1}(i) a_{ij}\} b_j(o_t) = \delta_{t-1}(\psi_t(j)) a_{\psi_t(j),j} b_j(o_t)$
  Fin Pour
Fin Pour
$q_T^* = \arg \max_{1 \leq i \leq N} \{\delta_T(i)\}$
$P(V = O, S = Q^*) = \max_{1 \leq i \leq N} \{\delta_T(i)\} = \delta_T(q_T^*)$
Pour $t = T-1$ **à** 1 **Faire**
  $q_t^* = \psi_{t+1}(q_{t+1}^*)$
Fin Pour

Tout comme pour les algorithmes *Forward* et *Backward*, cet algorithme souffre de problèmes liés à l'implémentation. Pour les résoudre, il est également nécessaire de mettre en place une stratégie de ré-échelonnement. A cet effet, on définit

$$\tilde{\delta}_t(i) = \max_{(q_1,\ldots,q_{t-1}) \in \mathbb{S}^{t-1}} \ln P(S_1 = q_1, \ldots, S_{t-1} = q_{t-1}, S_t = s_i, V_1 = o_1, \ldots, V_t = o_t / \lambda)$$

L'algorithme de Viterbi avec ré-échelonnement (Slimane, 2002) est alors donné par l'algorithme 1.6. Sa complexité est $O(N^2T)$, cependant, le calcul des logarithmes peut s'avérer plus coûteux.

Algorithme 1.6: Algorithme de Viterbi avec ré-échelonnement

Pour $i = 1$ **à** N **Faire**
  $\tilde{\delta}_1(i) = \ln \pi_i + \ln b_i(o_1)$
Fin Pour
Pour $t = 2$ **à** T **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\tilde{\psi}_t(j) = \arg \max_{1 \leq i \leq N} \left\{\tilde{\delta}_{t-1}(i) + \ln a_{ij}\right\}$
    $\tilde{\delta}_t(j) = \max_{1 \leq i \leq N} \left\{\tilde{\delta}_{t-1}(i) + \ln a_{ij}\right\} + \ln b_j(o_t) = \tilde{\delta}_{t-1}(\tilde{\psi}_t(j)) + \ln a_{\tilde{\psi}_t(j),j} + \ln b_j(o_t)$
  Fin Pour
Fin Pour
$q_T^* = \arg \max_{1 \leq i \leq N} \left\{\tilde{\delta}_T(i)\right\}$
$\ln P(V = O, S = Q^*) = \max_{1 \leq i \leq N} \left\{\tilde{\delta}_T(i)\right\} = \tilde{\delta}_T(q_T^*)$
Pour $t = T-1$ **à** 1 **Faire**
  $q_t^* = \tilde{\psi}_{t+1}(q_{t+1}^*)$
Fin Pour

1.6 Apprentissage des modèles de Markov cachés

Apprendre un MMC c'est ajuster les paramètres du modèle de manière à maximiser un certain critère. Différents critères sont disponibles dans la littérature. Nous n'allons pas tous les recenser, mais nous allons présenter les plus importants et les plus couramment utilisés.

1.6.1 Apprentissage étiqueté

Pour effectuer un apprentissage étiqueté, également connu dans la littérature comme l'apprentissage de Viterbi, on dispose de deux informations : la séquence d'observations O et la séquence d'états cachés Q qui a engendré la séquence précédente. Le critère que l'on cherche à maximiser est $P(V = O, S = Q/\lambda)$. Pour le maximiser, il suffit de compter les différentes transitions du système. Habituellement, avec ce type d'apprentissage, on ne considère pas une seule séquence d'observations à la fois, mais plusieurs. Nottons $\{O^1, \ldots, O^K\}$ les séquences d'observations, $\{Q^1, \ldots, Q^K\}$ les séquences d'états associées et $\{T^1, \ldots, T^K\}$ les longueurs des séquences. Dans ce cas, on utilise toujours le comptage des différentes transitions du système, mais en considérant toutes les séquences simultanément de manière indistincte. L'algorithme d'apprentissage étiqueté est donné par l'algorithme 1.7. Sa complexité est $O(N^2 + NM + T)$ en désignant par T, la longueur totale des séquences d'observations considérées.

Algorithme 1.7: Apprentissage étiqueté

$\forall\ i = 1..N,\ x_i = 0$
$\forall\ i = 1..N,\ j = 1..N,\ y_{i,j} = 0$
$\forall\ i = 1..N,\ j = 1..M,\ z_{i,j} = 0$
Pour $k = 1$ **à** K **Faire**
  Incrémenter $x_{q_1^k}$
  Pour $t = 1$ **à** T^k **Faire**
    Incrémenter $y_{q_t^k, o_t^k}$
    Si $t < T^k$ **Alors**
      Incrémenter $z_{q_t^k, q_{t+1}^k}$
    Fin Si
  Fin Pour
Fin Pour
$\forall\ i = 1..N,\ \pi_i = \frac{x_i}{\sum_{r=1}^{N} x_r}$
$\forall\ i = 1..N,\ j = 1..N,\ a_{i,j} = \frac{y_{i,j}}{\sum_{r=1}^{N} y_{i,r}}$
$\forall\ i = 1..N,\ j = 1..M,\ b_i(j) = \frac{z_{i,j}}{\sum_{r=1}^{M} z_{i,r}}$

Lorsque le nombre de séquences d'observations ou de séquences d'états cachés ou tout simplement le nombre d'apparitions d'un ou plusieurs motifs est trop réduit, l'apprentissage est souvent peu efficace, car le modèle n'arrive pas à généraliser ce qu'il doit reconnaître. En effet, de nombreuses probabilités sont très petites, voire nulles. Un moyen de résoudre ces problèmes consiste à effectuer un lissage lors de l'estimation des probabilités. En notant $c > 0$ le coefficient de lissage, l'algorithme est donné par l'algorithme 1.8. Dans cet algorithme, le coefficient de lissage est identique pour toutes les probabilités, mais rien n'empêche de le choisir différent pour chacune d'elles, afin d'inclure des connaissances expertes dans l'apprentissage.

Algorithme 1.8: Apprentissage étiqueté avec lissage

$\forall\ i = 1..N,\ x_i = 0$
$\forall\ i = 1..N,\ j = 1..N,\ y_{i,j} = 0$
$\forall\ i = 1..N,\ j = 1..M,\ z_{i,j} = 0$
Pour $k = 1$ **à** K **Faire**
  Incrémenter $x_{q_1^k}$
  Pour $t = 1$ **à** T^k **Faire**
    Incrémenter $y_{q_t^k, o_t^k}$
    Si $t < T^k$ **Alors**
      Incrémenter $z_{q_t^k, q_{t+1}^k}$
    Fin Si
  Fin Pour
Fin Pour
$\forall\ i = 1..N,\ \pi_i = \frac{c + x_i}{Nc + \sum_{r=1}^{N} x_r}$
$\forall\ i = 1..N,\ j = 1..N,\ a_{i,j} = \frac{c + y_{i,j}}{Nc + \sum_{r=1}^{N} y_{i,r}}$
$\forall\ i = 1..N,\ j = 1..M,\ b_i(j) = \frac{c + z_{i,j}}{Mc + \sum_{r=1}^{M} z_{i,r}}$

1.6.2 Maximisation de la vraisemblance

La critère de maximum de vraisemblance consiste à trouver le modèle λ^* maximisant la probabilité $P(V = O/\lambda)$ (Rabiner, 1989). En général, il n'est pas possible de trouver ce modèle optimal. Néanmoins, pour tenter de résoudre ce problème, il existe principalement deux méthodes : utiliser l'algorithme *Expectation-Maximization*, ou utiliser une descente de gradient.

1.6.2.1 Introduction à l'algorithme *Expectation-Maximization*

L'algorithme *Expectation-Maximization* (EM) est une méthode générale d'optimisation en présence d'information incomplète. L'algorithme permet, à partir d'un modèle initial m', de trouver un modèle m qui augmente la vraisemblance. Dans cette section, nous ne démontrerons pas l'algorithme EM, nous nous contenterons juste d'exposer les principes et formules qui nous seront nécessaires par la suite. Le lecteur intéressé trouvera dans (Dempster et al., 1977) un exposé plus complet de la méthodologie de l'algorithme *Expectation-Maximization*. Particulièrement bien adapté à des probabilités, l'algorithme EM repose sur deux hypothèses simples :

- maximiser $P(X = x/M = m)$ est équivalent à maximiser $\ln P(X = x/M = m)$;
- l'introduction de variables non observées ou cachées Y définies sur $\mathbb{Y}$ dans l'expression de la vraisemblance permet d'effectuer les calculs plus facilement.

Dans le cas de variables aléatoires discrètes, on définit $\Gamma_x(m, m')$ (Amini, 2001) par :

$$\begin{aligned}\Gamma_x(m, m') &= \sum_{y \in \mathbb{Y}} P(Y = y/X = x, M = m') \ln P(X = x, Y = y/M = m) \\ &= E_{y \in \mathbb{Y}} \left[\ln P(X = x, Y = y/M = m)/X = x, M = m' \right]\end{aligned}$$

avec $E_{\mathbb{Y}}[f]$ l'espérance mathématique de f sur l'ensemble $\mathbb{Y}$.

L'algorithme EM (Dempster et al., 1977) (Moon, 1996) consiste donc à construire, à partir d'un modèle initial m_1, une suite de modèles $(m_t)_{t>0}$ vérifiant

$$\Gamma_x(m_{t+1}, m_t) \geq \Gamma_x(m_t, m_t)$$

Une condition suffisante est alors de rechercher le modèle m_{t+1} qui maximise la fonction $\Gamma_x(m_{t+1}, m_t)$. La suite $(m_t)_{t>0}$ vérifie, pour tout $t > 1$ et $m_{t+1} \neq m_t$, la relation

$$P(X = x/M = m_{t+1}) > P(X = x/M = m_t)$$

L'une des plus célèbres applications de l'algorithme EM est l'algorithme Baum-Welch permettant l'apprentissage des MMC (Bilmes, 1998).

1.6.2.2 L'algorithme de Baum-Welch

Dans le cas des MMC, on cherche à maximiser $P(V = O/\lambda)$ où O désigne une séquence de T observations. En appliquant l'algorithme EM à la maximisation de cette probabilité (Bilmes, 1998), on est amené à maximiser $\Gamma(\lambda, \lambda')$, avec $\lambda = (A, B, \Pi)$ le nouveau modèle et λ' le modèle connu (ou actuel) :

$$\Gamma_O(\lambda, \lambda') = \sum_{Q \in \mathbb{S}^T} P(S = Q/V = O, \lambda') \ln P(V = O, S = Q/\lambda)$$

En effectuant les différents calculs (cf. annexe A), on obtient :

$$\begin{aligned}
\pi_i &= P(S_1 = s_i/O, \lambda') \\
a_{i,j} &= \frac{\sum_{t=1}^{T-1} P(S_t = s_i, S_{t+1} = s_j/V = O, \lambda')}{\sum_{t=1}^{T-1} P(S_t = s_i/V = O, \lambda')} \\
b_i(j) &= \frac{\sum_{t=1}^{T} P(S_t = s_i/V = O, \lambda')\delta(o_t = j)}{\sum_{t=1}^{T} P(S_t = s_i/V = O, \lambda')}
\end{aligned}$$

Les formules de ré-estimation obtenues ci-dessus peuvent s'interpréter de la façon suivante :

$$\pi_i = \text{probabilité d'être dans l'état } s_i \text{ à l'instant } t = 1$$

$$a_{i,j} = \frac{\text{nombre de transitions de l'état } s_i \text{ vers l'état } s_j}{\text{nombre de fois où l'on quitte l'état } s_i}$$

$$b_j(k) = \frac{\text{nombre d'apparitions simultanées de l'état } s_j \text{ et du symbole } v_k}{\text{nombre d'apparitions de l'état } s_j}$$

On peut alors remarquer que le principe reste similaire à celui de l'apprentissage étiqueté de la section 1.6.1, à la différence que « l'étiquetage » s'effectue en probabilité avant ré-estimation.

L'algorithme de Baum-Welch (Baum and Eagon, 1967) est donné par l'algorithme 1.9. Sa complexité est $O(N^2T + NMT)$.

D'une manière naïve, les probabilités utilisées pour la ré-estimation des matrices peuvent être obtenues par les algorithmes *Forward* et *Backward*. Cependant, toujours pour des problèmes d'implémentation numériques, on utilise plutôt leurs versions utilisant les algorithmes avec ré-échelonnement (cf. équations 1.1 et 1.2).

Algorithme 1.9: Algorithme de Baum-Welch

Choisir un modèle initial λ_0
$t = 0$
Répéter
  $t \leftarrow t + 1$
  Calculer les variables *Forward* et *Backward* pour le modèle λ_{t-1}
  Calculer Π de λ_t
  Calculer A de λ_t
  Calculer B de λ_t
Tant que $(P(V = O/\lambda_t) > P(V = O/\lambda_{t-1}))$ **et** $(t < t_{\max})$

1.6.2.3 Descente de gradient

La deuxième méthode permettant d'optimiser la vraisemblance $P(V = O/\lambda)$ consiste à utiliser la descente de gradient (Ganapathiraju, 1999). L'utilisation de la descente de gradient avec des MMC pose un problème de taille : les contraintes de stochasticité doivent être respectées par les paramètres du modèle.

Changement de l'espace de représentation

Une solution simple consiste à re-paramétrer les MMC avec des variables prenant leurs valeurs dans l'espace réel $\mathbb{R}$ à l'aide des équations suivantes :

$$\forall\ 1 \leq i,j \leq N, \qquad a_{i,j} = \frac{\exp(x_{i,j})}{\sum_{k=1}^{N} \exp(x_{i,k})}$$

$$\forall\ 1 \leq i \leq N,\ 1 \leq j \leq M, \qquad b_i(j) = \frac{\exp(y_{i,j})}{\sum_{k=1}^{M} \exp(y_{i,k})}$$

$$\forall\ 1 \leq i \leq N, \qquad \pi_i = \frac{\exp(z_i)}{\sum_{k=1}^{N} \exp(z_k)}$$

Si l'on suppose que les coefficients des matrices stochastiques sont strictement positifs, alors il existe au moins une solution à ces équations. Un MMC λ est alors parfaitement défini par les trois matrices stochastiques $\lambda = (A, B, \Pi)$ ou les trois matrices réelles $\lambda = (X, Y, Z)$. Dans le cas où tous les coefficients ne sont pas strictement positifs, il est toujours possible de fixer le coefficient nul à une valeur très petite, mais non nulle, de manière à ne pas trop déformer le modèle (cf. ci-dessous).

Il est intéressant de noter que les paramètres $x_{i,j}$, $y_{i,j}$ et z_i ne sont pas uniques. En effet, l'ajout d'une constante commune à chaque bloc de variables stochastiques donne des valeurs vérifiant également les équations. Pour passer des coefficients $a_{i,j}$, $b_i(j)$ et π_i aux coefficients $x_{i,j}$, $y_{i,j}$ et z_i, il suffit alors d'utiliser les formules suivantes :

$$x_{i,j} = \ln a_{i,j} \tag{1.3}$$

$$y_{i,j} = \ln b_i(j) \tag{1.4}$$

$$z_i = \ln \pi_i \tag{1.5}$$

L'utilisation de ce paramétrage pour le calcul des dérivées partielles pose cependant problème lorsque l'une des probabilités du modèle est nulle. Une solution couramment utilisée est

d'imposer que les coefficients soient non nuls. Une autre solution consiste à définir l'opérateur LN en imposant une valeur $\epsilon > 0$ proche de zéro et une valeur HV négative et grande en valeur absolue, telle que

$$LN(x) = \begin{cases} HV & \text{si } x \leq \epsilon \\ \ln(x) & \text{sinon} \end{cases}$$

En remplaçant l'opérateur ln dans les équations 1.3, 1.4 et 1.5, il est possible de ne pas imposer de contraintes de stricte positivité aux coefficients du modèle. Cependant, il faut avoir à l'esprit que cette transformation n'est pas réversible.

Calcul du gradient

Soit $L(\lambda) = \ln P(V = O/\lambda)$. Maximiser la vraisemblance $P(V = O/\lambda)$ est équivalent à maximiser le logarithme de la vraisemblance $L(\lambda)$. Calculons le gradient de $L(\lambda)$ par rapport aux paramètres de λ au point μ.

$$\frac{\partial L(\lambda)}{\partial \lambda}(\mu) = \frac{1}{P(V = O/\mu)} \frac{\partial P(V = O/\lambda)}{\partial \lambda}(\mu)$$

Maximiser $L(\lambda)$ nécessite donc de calculer les dérivées partielles de $P(V = O/\lambda)$ par rapport aux différents paramètres du modèle. Les calculs de l'annexe B nous permettent d'écrire :

$$\begin{aligned}
\frac{\partial P(V = O/\lambda)}{\partial x_{i,j}} &= \sum_{t=2}^{T} \alpha_{t-1}(i) a_{i,j} b_j(o_t) \beta_t(j) - a_{i,j} \sum_{t=2}^{T} \alpha_{t-1}(i) \beta_{t-1}(i) \\
\frac{\partial P(V = O/\lambda)}{\partial y_{i,j}} &= \sum_{t=1}^{T} \beta_t(i) \alpha_t(i) (\kappa(o_t = j) - b_i(j)) \\
\frac{\partial P(V = O/\lambda)}{\partial z_i} &= \pi_i b_i(o_1) \beta_1(i) - \pi_i P(V = O/\lambda)
\end{aligned}$$

En utilisant les variables *Forward* et *Backward* avec ré-échelonnement, on obtient

$$\begin{aligned}
\frac{1}{P(V = O/\lambda)} \frac{\partial P(V = O/\lambda)}{\partial x_{i,j}} &= \sum_{t=2}^{T} \tilde{\alpha}_{t-1}(i) a_{i,j} b_j(o_t) \tilde{\beta}_t(j) - a_{i,j} \sum_{t=2}^{T} \frac{\tilde{\alpha}_{t-1}(i) \tilde{\beta}_{t-1}(i)}{c_{t-1}} \\
\frac{1}{P(V = O/\lambda)} \frac{\partial P(V = O/\lambda)}{\partial y_{i,j}} &= \sum_{t=1}^{T} \frac{\tilde{\beta}_t(i) \tilde{\alpha}_t(i) (\kappa(o_t = j) - b_i(j))}{c_t} \\
\frac{1}{P(V = O/\lambda)} \frac{\partial P(V = O/\lambda)}{\partial z_i} &= \pi_i b_i(o_1) \tilde{\beta}_1(i) - \pi_i
\end{aligned}$$

A partir de ce gradient, il est possible d'utiliser n'importe quelle descente de gradient telle que celles décrites dans (Kapadia, 1998) et (Ganapathiraju, 1999). Cependant, il faut garder à l'esprit que ce calcul est coûteux en temps machine. Sa complexité est $O(N^2T + NMT)$.

1.6.2.4 Remarques

Que ce soit l'algorithme de Baum-Welch ou la descente de gradient, les deux méthodes nécessitent un modèle initial à améliorer. Ces approches simples possèdent un gros inconvénient : elles sont sensibles au point de départ et elles convergent vers des optima locaux de la vraisemblance. Nous verrons dans les chapitres suivants qu'il est nécessaire de mettre en

place des stratégies de recherche et d'exploration pour ne pas être piégé dans ces optima. Il existe de nombreuses variantes de l'algorithme EM construites pour pallier à certains de ces inconvénients.

Stochastic Expectation Maximization

L'algorithme *Stochastic Expectation Maximization* (SEM) (Celeux and Diebolt, 1986) peut également être utilisé pour effectuer l'apprentissage de MMC. L'algorithme SEM est une variante stochastique de l'algorithme EM, beaucoup moins sensible au point de départ. À partir d'un modèle initial λ_0, il consiste à engendrer en probabilité, selon la loi de λ_0, une séquence d'états cachés ayant engendré la séquence d'observations. À partir de cette séquence d'états cachés, un apprentissage étiqueté (cf. section 1.6.1) est réalisé afin d'obtenir un nouveau modèle λ_1. La procédure est réitérée plusieurs fois.

Cette méthode possède deux avantages importants par rapport à l'algorithme EM (Celeux and Diebolt, 1989) : la convergence est rapide et l'algorithme SEM est peu sensible au modèle initial. Cependant, la méthode possède également deux gros désavantages : elle est moins efficace que l'algorithme EM (Baum-Welch) en présence de séquences d'observations trop courtes et la suite des modèles obtenus ne converge pas ponctuellement (on n'a pas $P(V = O/\lambda_n) \leq P(V = O/\lambda_{n+1})$ mais uniquement globalement.

Le lecteur pourra remarquer que cet algorithme est proche de l'algorithme de *segmental k-means* de la section 1.6.5 : seul le mode de génération de la séquence d'états cachés change.

Estimation Conditionnelle Itérative : ICE

L'algorithme d'Estimation Conditionnelle Itérative (*Iterative Conditional Estimation* ou ICE) est une méthode d'optimisation en présence de données cachées proposée dans (Pieczynski, 1990). Le principe de ICE consiste à utiliser un estimateur des paramètres du modèle calculé à partir des informations complétées, c'est-à-dire à partir de la séquence d'observations et d'une séquence d'états cachés. Il a été montré que la meilleure approximation du modèle au sens de l'erreur quadratique moyenne est l'espérance conditionnelle. Ainsi, dans le cas particulier des MMC considérés (Brouard, 1999), l'algorithme ICE permet d'aboutir aux mêmes formules de ré-estimation que l'algorithme de Baum-Welch.

Les autres variantes

D'autres variantes de l'algorithme EM sont disponibles dans la littérature, mais elles sont peu adaptées ou peu utilisées pour les MMC que nous considérons. Il est possible de citer l'algorithme de SAEM (Celeux and Diebolt, 1989), qui est un intermédiaire entre EM et SEM, ou MCEM (Cappé et al., 1999), qui utilise de manière intense la génération de séquences d'états et les méthodes de Monte-Carlo.

1.6.3 Critère du maximum *a posteriori* (MAP)

Le critère de maximum *a posteriori* (MAP) trouve son intérêt dans la théorie de la décision bayésienne. Jusqu'à maintenant, nous avons considéré des critères d'optimisation des modèles utilisant la règle de décision suivante :

Si $P(V = O/\lambda_1) > P(V = O/\lambda_2)$ alors O a été le plus probablement émise par le modèle λ_1

Bien que la notion de « plus probablement émise » soit couramment utilisée afin de dire « appartient », c'est-à-dire, pour notre exemple, que la séquence d'observations O appartient à la classe modélisée par λ_1, rien ne garantit que ce choix soit optimal.

Un moyen de garantir un choix optimal, au moins en théorie, est d'utiliser la théorie de la décision bayésienne (Berthold and Hand, 1998). Le critère de décision utilisé est alors

$$\text{Si } P(\lambda_1/V=O) > P(\lambda_2/V=O), \text{ alors } O \text{ appartient à la classe } \lambda_1$$

Ce critère pose problème, car nous ne savons pas comment exprimer ces probabilités. Cependant, en transformant ces probabilités, on obtient

$$P(\lambda/V=O) = \frac{P(V=O/\lambda)P(\lambda)}{P(V=O)}$$

Que signifient ces probabilités ? $P(\lambda/V=O)$ est la probabilité *a posteriori* du modèle λ connaissant la séquence d'observations O, $P(\lambda)$ est la probabilité *a priori* d'apparition du modèle λ et $P(V=O)$ est la probabilité *a priori* d'apparition de la séquence d'observations O.

Le critère MAP possède un avantage certain sur le critère de maximum de vraisemblance : les probabilités *a priori* permettent de modéliser le déséquilibre éventuel dans l'apparition des séquences d'observations.

La première remarque que l'on peut faire est que les probabilités $P(V=O)$ peuvent être ignorées car, dans la règle de décision bayésienne, elles peuvent être simplifiées.

L'apprentissage des modèles avec le critère MAP dépend très fortement des objectifs visés. Lorsque les modèles sont appris séparément l'objectif est alors de maximiser la probabilité $P(\lambda/V=O)$, c'est-à-dire, après simplification ,maximiser $P(V=O/\lambda)P(\lambda)$. Si la probabilité $P(\lambda)$ s'exprime indépendamment des valeurs prises par les matrices stochastiques qui le définissent, alors les deux probabilités peuvent être apprises séparément. Pour la probabilité $P(V=O/\lambda)$, il suffit d'utiliser le critère de maximum de vraisemblance et pour la probabilité $P(\lambda)$ on utilise généralement une estimation statistique de l'apparition de ce modèle. Dans le cas où l'expression de $P(\lambda)$ dépend des valeurs prises par les paramètres du modèle, il n'est pas possible d'utiliser le critère de maximum de vraisemblance. Une solution consiste alors à utiliser une descente de gradient (cf. section 1.6.2.3) afin de maximiser le critère $\ln P(V=O/\lambda) + \ln P(\lambda)$ à condition que $P(\lambda)$ soit différentiable. Lorsque le critère devient plus complexe, ou lorsqu'il utilise plusieurs modèles ou séquences d'observations en simultané, la même démarche peut être utilisée : s'il est possible d'optimiser séparément les deux types de probabilités, il faut les traiter séparément. Dans les cas où cela n'est pas possible, le moyen le plus courant d'effectuer l'apprentissage consiste à utiliser la descente de gradient. Pour certains critères, il n'est pas possible d'éliminer les probabilités $P(V=O)$, il est alors nécessaire de les modéliser et de les inclure dans la descente de gradient.

1.6.4 Maximisation de l'information mutuelle

L'un des buts principaux de l'apprentissage de MMC est d'effectuer une classification. En effet, on cherche souvent, à partir d'une observation O, à décider de manière automatique à quelle autre observation elle ressemble le plus et surtout à décider à quelle classe de séquences d'observations elle appartient réellement. Prenons un exemple. On considère un système d'identification biométrique basé sur la photographie du visage. Initialement, le système possède au moins une photographie de chacune des personnes à reconnaître. Chaque photographie est modélisée par un MMC après qu'elle ait été transformée par un procédé quelconque en séquence d'observations. Si une personne se présente devant la caméra, le système va prendre une photographie, la transformer en séquence et comparer les différentes vraisemblances avec les MMC appris. Le MMC qui permet d'obtenir la meilleure vraisemblance permet alors de dire que la personne est celle qui correspond à la photographie du MMC. En théorie, cette méthode fonctionne mais, en pratique, ce n'est pas toujours le cas. Si l'ensemble des photographies concerne des photographies de visages de personnes de même couleur de peau et de même couleur de cheveux, alors il y a de grandes chances pour que les

modèles reconnaissent bien l'ensemble des visages, car la modélisation des visages sera quasi identique. Une solution consiste à effectuer l'apprentissage des MMC avec un autre critère que la vraisemblance. Le critère de prédilection pour cette tâche est la maximisation de l'information mutuelle (MIM). Plusieurs variantes de la maximisation de l'information mutuelle existent : elles sont présentées ci-dessous.

1.6.4.1 Maximisation de l'information mutuelle de la vraisemblance

La première forme du critère de MIM s'attache à différencier les modèles par leurs vraisemblances. A cet effet, on cherche à maximiser la vraisemblance de la séquence d'observations à apprendre O mais également à minimiser la vraisemblance des séquences d'observations à ne pas reconnaître $N^1, \ldots, N^K$. L'avantage de ce critère est qu'il laisse le MMC modéliser ce qui est caractéristique, tout en acceptant de moins bien modéliser ce qui ne l'est pas.

Ce critère peut prendre plusieurs formes. La forme présentée ci-après est celle décrite dans (Rabiner, 1989). Cette forme est intéressante, car elle permet de gérer facilement les problèmes de précision numérique.

$$mim(\lambda) = \frac{P(V = O/\lambda)}{\prod_{k=1}^{K} P(V = N^k/\lambda)}$$

Comme nous pouvons le voir, maximiser cette expression entraîne la maximisation de la vraisemblance $P(V = O/\lambda)$ et la minimisation des vraisemblances $P(V = N^k/\lambda)$.

On remarque alors que maximiser $mim(\lambda)$ est équivalent à maximiser son logarithme népérien. On définit alors

$$\begin{aligned} L(\lambda) &= \ln mim(\lambda) \\ &= \ln P(V = O/\lambda) - \sum_{k=1}^{K} \ln P(V = N^k/\lambda) \end{aligned}$$

Pour optimiser ce critère, il est alors possible d'utiliser une descente de gradient. Il est donc nécessaire de calculer le gradient de $L(\lambda)$. Ce dernier est donné par l'équation suivante :

$$\frac{\partial L(\lambda)}{\partial \lambda} = \frac{1}{P(V = O/\lambda)} \frac{\partial P(V = O/\lambda)}{\partial \lambda} - \sum_{k=1}^{K} \frac{1}{P(V = N^k/\lambda)} \frac{\partial P(V = N^k/\lambda)}{\partial \lambda}$$

Or ce gradient n'est autre qu'une combinaison linéaire des gradients calculés à la section 1.6.2.3. Pour cela, on note $\tilde{\alpha}_t(i)$, $\tilde{\beta}_t(i)$, c_t les variables *Forward* et *Backward* avec ré-échelonnement et les coefficients de ré-échelonnement calculés pour la séquence d'observation O. On note $\tilde{\alpha}^k_t(i)$, $\tilde{\beta}^k_t(i)$ et c^k_t les variables *Forward* et *Backward* avec ré-échelonnement et les coefficients de ré-échelonnement calculés pour la séquence d'observations N^k. Alors, en reprenant les équations de la section précédente, on obtient

$$\begin{aligned} \frac{\partial L(\lambda)}{\partial x_{i,j}} &= \sum_{t=2}^{T} \tilde{\alpha}_{t-1}(i) a_{i,j} b_j(o_t) \tilde{\beta}_t(j) - a_{i,j} \sum_{t=2}^{T} \frac{\tilde{\alpha}_{t-1}(i)\tilde{\beta}_{t-1}(i)}{c_{t-1}} \\ &\quad - \sum_{k=1}^{K} \left(\sum_{t=2}^{T} \tilde{\alpha}^k_{t-1}(i) a_{i,j} b_j(n^k_t) \tilde{\beta}^k_t(j) - a_{i,j} \sum_{t=2}^{T} \frac{\tilde{\alpha}^k_{t-1}(i)\tilde{\beta}^k_{t-1}(i)}{c^k_{t-1}} \right) \end{aligned}$$

$$\begin{aligned} \frac{\partial L(\lambda)}{\partial y_{i,j}} &= \sum_{t=1}^{T} \frac{\tilde{\beta}_t(i)\tilde{\alpha}_t(i)(\kappa(o_t = j) - b_i(j))}{c_t} \\ &\quad - \sum_{k=1}^{K} \left(\sum_{t=1}^{T} \frac{\tilde{\beta}^k_t(i)\tilde{\alpha}^k_t(i)(\kappa(n^k_t = j) - b_i(j))}{c^k_t} \right) \end{aligned}$$

et

$$\frac{\partial L(\lambda)}{\partial z_i} = \pi_i b_i(o_1)\tilde{\beta}_1(i) - \pi_i - \sum_{k=1}^{K}\Big(\pi_i b_i(n_1^k)\tilde{\beta}_1^k(i) - \pi_i\Big)$$

Il suffit alors d'utiliser la technique de la descente de gradient.

1.6.4.2 Maximisation de l'information mutuelle du MAP

Le critère de maximisation de l'information mutuelle pour le critère de décision MAP décrit dans (Giurgiu, 2002) conduit à la minimisation du critère $I(\lambda, O)$ avec $\mathbb{L}$ l'ensemble des L MMC $\{\lambda_1, \ldots, \lambda_L\}$ représentant les L classes et $\mathbb{O}$ l'ensemble des séquences d'observations à apprendre. On définit $c(O_i) \in [1..L]$ le numéro de la classe associée à la séquence d'observations O_i. Le critère est :

$$I(\mathbb{L}, \mathbb{O}) = H(\mathbb{L}) - I(\mathbb{L}||\mathbb{O})$$

avec

$$H(\mathbb{L}) = -\sum_{\lambda_i \in \lambda} P(\lambda_i) \ln P(\lambda_i)$$

et

$$I(\mathbb{L}||\mathbb{O}) = \sum_{\lambda_i \in \lambda}\sum_{O_j \in O} P(V = O_j, \lambda_i) \ln \frac{P(V = O_j, \lambda_i)}{P(\lambda_i)P(V = O_j)}$$

Si l'on considère que les probabilités $P(V = O_i)$ sont constantes et que les probabilités $P(\lambda_i/V = O_j)$ sont nulles, sauf quand $i = j$, alors minimiser $I(\mathbb{L}, \mathbb{O})$ est équivalent à maximiser (Giurgiu, 2002) (Schluter et al., 1997) :

$$\tilde{I}(\mathbb{L}||\mathbb{O}) = \sum_{\lambda_i \in \lambda}\sum_{O_j \in C_i} \frac{P(V = O_j/\lambda_i)P(\lambda_i)}{\sum\limits_{\lambda_l \in \lambda} P(V = O_j/\lambda_i)P(\lambda_i)}$$

avec $C_i = \{O_j \in O / c(O_j) = i$.

La maximisation de $\tilde{I}(\mathbb{L}||\mathbb{O})$ peut alors être réalisée grâce à une descente de gradient ou grâce à l'algorithme de Baum-Welch étendu, dont les formules de ré-estimation sont décrites dans (Schluter et al., 1997). Le lecteur intéressé pourra noter que (Vertanen, 2004) présente une autre forme de MMI.

1.6.5 Le critère de *segmental k-means*

Parmi l'ensemble des critères utilisés pour l'apprentissage de MMC, le critère de *segmental k-means* se détache des autres. En effet, pour ce critère, on cherche à optimiser la probabilité $P(V = O, S = Q^*/\lambda)$ avec Q^* la séquence d'états cachés qui a le plus probablement engendré la séquence telle que calculée par l'algorithme de Viterbi (cf. section 1.5.2) (Juang and Rabiner, 1990) (Dugad and Desai, 1996). Une des grandes difficultés de ce critère est qu'il n'est ni dérivable, ni même continu. Par conséquent, les méthodes s'appuyant sur l'algorithme EM ou les descentes de gradient ne sont pas utilisables. Cependant, il existe quand même un moyen d'ajuster les paramètres d'un modèle de manière à maximiser cette probabilité. Cet algorithme appelé *segmental k-means* repose sur deux algorithmes décrits précédemment : l'algorithme de Viterbi et l'apprentissage étiqueté.

Son principe est simple :
- à partir d'un modèle initial et de la séquence d'observations O, on recherche la séquence d'états cachés qui a le plus probablement été suivie pour générer O à l'aide de l'algorithme de Viterbi. Cette recherche permet d'étiqueter la séquence O et par conséquent de la segmenter ;
- une fois étiquetée, la séquence O est alors apprise par comptage des transitions effectives entre les états et les émissions de symboles. Cette étape peut alors être considérée comme un *k-means* consistant à ré-estimer les « centres des classes » ;
- le nouveau modèle est alors utilisé comme modèle initial et les deux opérations précédentes sont répétées tant que nécessaire.

Il a pu être montré (Juang and Rabiner, 1990) que l'algorithme de *segmental k-means* (cf. algorithme 1.10) permettait d'augmenter la probabilité $P(V = O, S = Q^*/\lambda)$ de manière itérative et qu'il convergeait vers un maximum local du critère considéré. Lorsque l'on utilise ce critère, il faut faire attention à sa formulation. L'augmentation itérative de la probabilité consiste à trouver un modèle λ_{k+1} à partir d'un modèle λ_k tel que $P(V = O, S = Q_k^*/\lambda_k) < P(V = O, S = Q_{k+1}^*/\lambda_{k+1})$ et non pas tel que $P(V = O, S = Q_k^*/\lambda_k) < P(V = O, S = Q_k^*/\lambda_{k+1})$ avec Q_k^* la séquence de Viterbi obtenue avec le modèle λ_k et Q_{k+1}^* la séquence de Viterbi obtenue avec le modèle λ_{k+1}. En effet, la séquence de Viterbi change lorsque le modèle est modifié.

Algorithme 1.10: Algorithme de *segmental k-means*

Choisir un MMC initial λ_O
$t = 0$
Répéter
  $t = t + 1$
  $Q^* = \text{Viterbi}(O, \lambda_{t-1})$
  Estimer λ_t à partir de O et Q^*
Tant que $P(V = O, S = Q_t^*/\lambda_t) > P(V = O, S = Q_{t-1}^*/\lambda_{t-1})$

L'algorithme de *segmental k-means* peut également être utilisé avec plusieurs séquences d'observations. Pour cela, il suffit de considérer le critère d'apprentissage

$$\prod_{k=1}^{K} P(V = O^k, S = Q^{k*}/\lambda)$$

L'algorithme consiste alors à appliquer l'algorithme de Viterbi à chacune des séquences d'observations et à utiliser l'apprentissage étiqueté de toutes ces séquences simultanément, comme décrit à la section 1.6.1.

Cet algorithme est parfois utilisé en raison de sa rapidité en lieu et place de l'algorithme de Baum-Welch, en considérant l'hypothèse suivante : les probabilités complétées $P(V = O, S = Q/\lambda)$ sont nulles ou négligeables pour toutes les séquences d'états, à l'exception de celle de la séquence Q^* de Viterbi associée. Par conséquent, maximiser $P(V = O/\lambda)$ est équivalent à maximiser $P(V = O, S = Q^*/\lambda)$. Bien qu'il soit possible de trouver des modèles pathologiques contredisant cette hypothèse, dans la pratique, l'hypothèse est souvent confirmée.

1.6.6 Minimisation du taux d'erreur de classification

Ce critère a pour objectif de minimiser le taux d'erreur de classification avec une décision soumise au critère MAP. Pour le décrire brièvement, on considère un ensemble $\mathcal{E} = \{(O_i, c(O_i))\}$ de séquences d'observations ainsi que le numéro de la classe qui leur sont associés. Soit $\{1, \ldots, L\}$ les L classes à apprendre et $\{\lambda_1, \ldots, \lambda_L\}$ les MMC associés.

1.6.6.1 Première approche

Dans cette première approche décrite dans (Ljolje et al., 1990), les auteurs ont abouti au critère suivant, après plusieurs transformations et approximations :

$$\tau_e = 1 - \frac{1}{N} \sum_{m=1}^{L} \sum_{O_i \in \mathcal{E}, c(O_i)=m} \frac{P(V = O_i/\lambda_m)P(\lambda_m)}{\sum_{l=1}^{L} P(V = O_i/\lambda_l)P(\lambda_l)}$$

Si l'on suppose que les classes apparaissent avec la même probabilité *i.e.* $P(\lambda_i) = \frac{1}{L}$ pour tout $i = 1..L$, alors minimiser τ_e revient à maximiser $\bar{\tau}_e$ avec

$$\bar{\tau}_e = \frac{1}{N} \sum_{m=1}^{L} \sum_{O_i \in \mathcal{E}, c(O_i)=m} \frac{P(V = O_i/\lambda_m)}{\sum_{l=1}^{L} P(V = O_i/\lambda_l)}$$

Il suffit alors d'utiliser une descente de gradient sur l'ensemble des paramètres $\{\lambda_1, \ldots, \lambda_L\}$ pour maximiser $\bar{\tau}_e$ et donc minimiser τ_e.

Pour que l'approximation effectuée soit valable, il est nécessaire que les modèles initiaux utilisés par la descente de gradient aient été obtenus par la maximisation de $\prod_{O_i \in \mathcal{E}} P(V = O_i/\lambda_{c(O_i)})$ par un des algorithmes de maximisation de la vraisemblance décrit précédemment.

Bien que ce critère semble intéressant, il n'obtient pas toujours de bons résultats. En effet, minimiser τ_e ne garantit aucunement que ce taux sera faible sur un ensemble d'observations autre que celui utilisé pour l'apprentissage.

1.6.6.2 Deuxième approche

Dans (Saul and Rahim, 2000), le critère de minimisation des erreurs de classification (*minimum classification error* MCE) est défini sous la forme

$$J = \frac{1}{|E|} \sum_{O_i \in O} \Delta \left(\max_{\lambda_j \in \lambda, j \neq c(O_i)} \ln \frac{P(V = O_i/\lambda_j)}{P(V = O_i/\lambda_{c(O_i)})} \right)$$

avec $\Delta(z) = \begin{cases} 1 & \text{si } z \geq 1 \\ 0 & \text{sinon} \end{cases}$.

Ce critère n'est pas dérivable, ni même continu. Pour l'approcher sous forme continue, il suffit d'utiliser une sigmoïde $[1 + e^{-z}]^{-1}$ à la place de $\Delta(z)$ et l'opérateur *softmax* $\ln \sum_k e^{z_k}$ à la place de $\max_k z_k$. Le critère J peut alors être approché par (Saul and Rahim, 2000) :

$$\tilde{J} = \frac{1}{|E|} \sum_{O_i \in O} \frac{1}{1 + \exp\left(\ln P(V = O_i/\lambda_{c(O_i)}) - \ln \sum_{\lambda_j \in \lambda, j \neq c(O_i)} P(V = O_i/\lambda_j) \right)}$$

Le critère $\tilde{J}$ peut alors être minimisé à l'aide d'une descente de gradient.

Les deux formes de critères de minimisation des erreurs de classification évoquées précédemment ne sont pas les seules qui existent. Dans (Ganapathiraju, 1999), une autre forme est présentée.

1.6.7 Remarques générales sur les critères d'apprentissage

Comme nous venons de le voir, de nombreux critères peuvent être considérés pour l'apprentissage de modèles de Markov cachés. Les critères que nous avons décrits dans ce chapitre ne sont pas les seuls envisageables, mais ce sont les plus couramment utilisés. De plus, la démonstration des algorithmes d'apprentissage fournit la majorité des outils nécessaires à la conception des algorithmes d'apprentissage pour tous les critères envisageables.

Tous les algorithmes d'apprentissage de ce chapitre n'ont pas la même complexité. Pour faciliter le choix à la fois du critère et de l'algorithme de résolution, nous avons construit le tableau 1.1.

Critère	Algorithme	Complexité d'une itération
$P(V=O,S=Q/\lambda)$	apprentissage étiqueté	N^2+NM+T
$\prod_{k=1}^{K} P(V=O_k,S=Q_k/\lambda)$	apprentissage étiqueté	$N^2+NM+\sum_{k=1}^{K} T_k$
$P(V=O/\lambda)$	Baum-Welch	N^2T+NMT
$\prod_{k=1}^{K} P(V=O_k/\lambda)$	Baum-Welch	$N^2\sum_{k=1}^{K} T_k+NM\sum_{k=1}^{K} T_k$
$P(V=O/\lambda)$	Gradient	N^2T+NMT
$\prod_{k=1}^{K} P(V=O_k/\lambda)$	Gradient	$N^2\sum_{k=1}^{K} T_k+NM\sum_{k=1}^{K} T_k$
$\frac{P(V=O/\lambda)}{\prod_{k=1}^{K} P(V=O_k/\lambda)}$	Information mutuelle	$N^2(T+\sum_{k=1}^{K} T_k)$ $+NM(T+\sum_{k=1}^{K} T_k)$
$P(\lambda/V=O)$	MAP simple	N^2T+NMT
$P(\lambda/V=O)$	MAP complexe	Potentiellement très élevée
$P(V=O,S=Q^*/\lambda)$	*segmental k-means*	N^2T+NM
$\prod_{k=1}^{K} P(V=O_k,S=Q_k^*/\lambda)$	*segmental k-means*	$N^2\sum_{k=1}^{K} T_k+NM$

TAB. 1.1 – Complexité associée aux algorithmes en fonction des critères optimisés. $\{O,O_1,\ldots,O_K\}$ est l'ensemble des séquences d'observations de longueurs $\{T,T_1,\ldots,T_k\}$. N est le nombre d'états cachés du MMC. M est le nombre de symboles du MMC.

Il est intéressant de remarquer que l'algorithme de *segmental k-means* peut être beaucoup plus rapide que l'algorithme de Baum-Welch. En effet, il est très courant que le nombre de symboles M soit beaucoup plus grand que le nombre des états cachés. Dans ces conditions, lorsque la longueur T de la séquence d'observations augmente, le terme dominant dans la complexité de l'algorithme de Baum-Welch est NMT tandis que pour l'algorithme de *segmental k-means*, ce terme dominant est N^2T. Par conséquent, pour $M>N$, l'algorithme de *segmental k-means* est plus rapide que l'algorithme de Baum-Welch lorsque la longueur de la séquence d'observations augmente.

L'algorithme de *segmental k-means* est donc parfois utilisé en lieu et place de l'algorithme de Baum-Welch, car plusieurs auteurs ont remarqué que la probabilité $P(V=O,S=Q^*/\lambda)$ est élevée par rapport aux autres chemins d'états et qu'une grande majorité des chemins ont une probabilité très faible, voire nulle. Par conséquent, certains travaux émettent l'hypothèse que $P(V=O/\lambda)\approx P(V=O,S=Q^*/\lambda)$.

Un autre point important est que la complexité du calcul du gradient est du même ordre que celle de l'algorithme de Baum-Welch. Cette propriété est intéressante car elle signifie, *a priori*, qu'il n'est pas forcément beaucoup plus coûteux d'utiliser des critères tels que la maximisation de l'information mutuelle ou le critère MAP simple. En effet, on remarque que la complexité de l'optimisation de ces critères est linéaire en fonction de la longueur totale des séquences d'observations impliquées. Cependant, la descente de gradient peut nécessiter d'effectuer ce calcul plusieurs fois avant d'améliorer un modèle et par conséquent l'approche par descente de gradient est considérée comme étant relativement coûteuse.

1.7 Conclusion

Pour l'ensemble des critères de ce chapitre, nous avons été à même de définir un algorithme permettant, à partir d'un modèle initial, de trouver un nouveau modèle augmentant le critère. Cependant, nous ne sommes pas capables, d'une manière générale, de trouver le ou les modèles maximisant ce critère. En effet, la grande majorité des algorithmes sont piégés par les optima locaux du critère. Une solution pour essayer de trouver de bons modèles consiste à utiliser des heuristiques pour explorer l'espace. Le chapitre 2 est consacré à la description des heuristiques les plus couramment utilisées avec des modèles de Markov cachés.

Chapitre 2

Métaheuristiques pour l'apprentissage de modèles de Markov cachés

2.1 Introduction

Dans ce chapitre, nous nous intéressons à la description des principales métaheuristiques qui ont été utilisées pour l'apprentissage de modèles de Markov cachés. L'objectif de ce chapitre n'étant pas de réaliser une introduction aux différentes métaheuristiques utilisées, ni de dresser un panorama exhaustif des métaheuristiques utilisables, nous renverrons le lecteur intéressé aux références proposées dans chacune des sections suivantes pour plus de détail sur ces métaheuristiques.

Une métaheuristique peut souvent être vue comme un processus itératif utilisant successivement les quatre éléments suivants (Fonlupt et al., 1999) :

- une solution courante ou un ensemble de solutions courantes ;
- un opérateur de voisinage ν, qui, appliqué à la solution courante ou à l'ensemble de solutions courantes, construit un ensemble de solutions candidates ;
- une règle de transition qui choisit une solution courante ou un ensemble de solutions courantes parmi les solutions candidates.
- un critère d'arrêt de la boucle.

Les métaheuristiques décrites ci-dessous peuvent être séparées en deux catégories : les métaheuristiques sans recherche locale et les métaheuristiques avec recherche locale. Une métaheuristique avec recherche locale[1] utilise un algorithme d'optimisation locale tel que l'algorithme de Baum-Welch ou une descente de gradient (cf. chapitre 1), tandis que les métaheuristiques sans recherche locale s'appuient exclusivement sur une exploration de l'espace de recherche. Lorsque le critère d'optimisation est changé, il est bien entendu nécessaire pour les métaheuristiques avec recherche locale d'adapter l'algorithme d'optimisation locale en conséquence.

Les métaheuristiques présentées dans ce chapitre cherchent à optimiser un critère particulier, tel que la vraisemblance ou la probabilité de la séquence d'états cachés la plus probablement suivie. Cependant, dans toutes les métaheuristiques que nous allons décrire, il est généralement possible d'utiliser un autre critère d'apprentissage que celui d'origine, sans remettre en cause l'algorithme lui même.

Afin de faciliter l'exposé et de le rendre plus général, nous avons adopté les notations suivantes :

- Λ est l'espace des MMC.
- $C : \Lambda \rightarrow \mathbb{R}$ est la fonction objectif à maximiser.

[1] On parle généralement d'hybridation d'une métaheuristique avec une recherche locale.

- optim : $\Lambda \rightarrow \Lambda$ est un opérateur d'optimisation locale compatible avec C *i.e.* $C(x) \leq C(\text{optim}(x))$. On pourra remarquer que optim peut être l'identité *i.e.* $\text{optim}(x) = x$.
- s^* est le meilleur MMC trouvé par la métaheuristique.
- $\mathcal{U}(S)$ est l'opérateur engendrant une solution uniformément dans S.

2.2 L'heuristique *Random*

La première et la plus simple des heuristiques est l'heuristique *Random*. Cette heuristique se contente d'engendrer uniformément des solutions et de conserver celle qui maximise le critère. L'algorithme 2.1 donne sa description. L'intérêt principal de cet algorithme est de servir de référence. En effet, si le nombre de MMC engendrés tend vers l'infini, on sait (Clerc, 2005b) que cet algorithme convergera vers une solution optimale. Cependant, cette convergence est lente et nécessite d'engendrer énormément de MMC. Par conséquent, pour avoir un intérêt, les métaheuristiques que nous employons doivent la surpasser en efficacité.

Algorithme 2.1: L'heuristique *Random*

$s^* = \mathcal{U}(\Lambda)$
Tant que condition d'arrêt non vérifiée **Faire**
  $s = \mathcal{U}(\Lambda)$
  Si $C(s) > C(s^*)$ **Alors**
    $s^* = s$
  Fin Si
Fin Tant que

Random peut servir de référence pour les algorithmes n'utilisant pas d'opérateur d'optimisation locale mais, dans le cas contraire, la comparaison n'est pas juste. Il est donc nécessaire de définir la métaheuristique *Random+optim* suivant les mêmes principes mais utilisant les MMC engendrés comme des points de départ pour l'opérateur d'optimisation locale. L'algorithme associé est donné par 2.2.

Algorithme 2.2: L'heuristique *Random+optim*

$s^* = \mathcal{U}(\Lambda)$
Tant que condition d'arrêt non vérifiée **Faire**
  $s = \text{optim}(\mathcal{U}(\Lambda))$
  Si $C(s) > C(s^*)$ **Alors**
    $s^* = s$
  Fin Si
Fin Tant que

2.3 La méthode du recuit simulé

2.3.1 Principe du recuit simulé

Le recuit simulé est une métaheuristique inspirée de la technique de recuit utilisée en métallurgie (Kirkpatrick et al., 1983) afin d'obtenir des cristaux de faible énergie interne. La technique consiste, à partir d'un mélange chauffé en l'état « visqueux », à abaisser la température du mélange. Cette baisse de température peut être effectuée de deux façons : rapidement ou lentement. Une baisse rapide réalisée grâce à la technique de la trempe aboutit à un mélange solide amorphe représentant un minimum local de l'énergie du mélange. En

utilisant une baisse lente et éventuellement des réchauffes partielles du mélange, on obtient un cristal correspondant à un minimum global de l'énergie. La transposition de ces principes physiques au domaine de l'optimisation a conduit au recuit simulé (cf. figure 2.1).

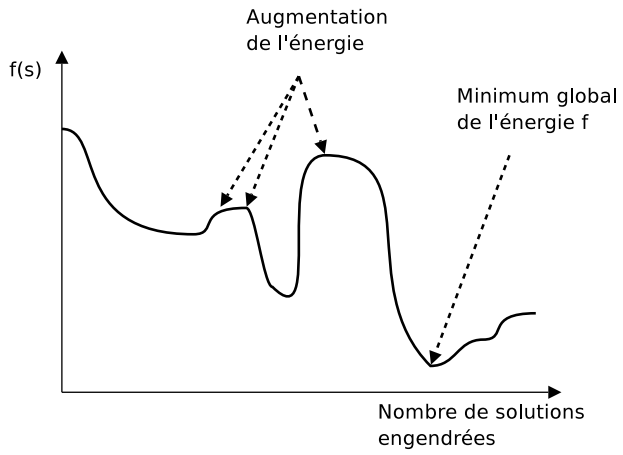


FIG. 2.1 – Exemple de comportement de la méthode du recuit simulé.

Les principes du recuit simulé ont été établis dès les années 1980 par S. Kirkpatrick, C. D. Gelatt et M. P. Vecchi dans (Kirkpatrick et al., 1983). L'algorithme de recuit simulé, dans le cas de la maximisation de f, est donné par l'algorithme 2.3.

Algorithme 2.3: Algorithme du recuit simulé

$s^* = \mathcal{U}(\mathcal{S})$
$s = s^*$
$T = T_0$
Tant que pas terminé **Faire**
  Construire une solution s' dans le voisinage de s
  Si $f(s') > f(s)$ **ou** $A(s, s', T) > \mathcal{U}([0; 1])$ **Alors**
    $s = s'$
  Fin Si
  Si $f(s) > f(s^*)$ **Alors**
    $s^* = s$
  Fin Si
  $T = U(T)$
Fin Tant que

A partir d'une solution initiale $s \in \mathcal{S}$, on engendre une solution voisine s'. La solution s' remplace la solution s si elle est acceptée selon une certaine règle. Habituellement, cette règle consiste à accepter s' si s' augmente la fonction objectif ou si s' ne dégrade « pas trop » la fonction objectif. La notion de « pas trop » dépend de la probabilité d'acceptation $A(s, s', T)$. Cette probabilité est fonction de la solution courante, de la solution candidate et de la température. Deux fonctions sont couramment utilisées (Locatelli, 2002) : la fonction de Metropolis $A(s, s', T) = \min\left\{1, \exp\left\{-\frac{f(s)-f(s')}{T}\right\}\right\}$ et la fonction de Baker $A(s, s', t) = \frac{1}{1+\exp\left\{\frac{f(s)-f(s')}{T}\right\}}$. Au fur et à mesure des essais, la température T est diminuée via la fonction U. U peut prendre de nombreuses formes, telles qu'une diminution géométrique ou par paliers. D'une manière générale, la fonction U et la température initiale T_0 sont spécifiques au problème.

Deux utilisations du recuit simulé ont été proposées pour l'apprentissage de MMC.

2.3.2 Première application du recuit simulé pour l'apprentissage de MMC

La première application de la technique du recuit simulé au problème de l'apprentissage de MMC a été réalisée par D. B. Paul (Paul, 1985). L'approche adoptée par l'auteur a été d'utiliser le recuit simulé sur les matrices stochastiques des modèles.

Dans cette application, l'auteur cherche à maximiser la vraisemblance du MMC. Pour cela, il suppose que $\pi_1 = 1$ et $\pi_i = 0$ pour tout $i \neq 1$. Le meilleur opérateur de perturbation d'une solution que D. B. Paul a obtenu consiste à appliquer la perturbation alternativement sur la matrice A ou sur la matrice B. La perturbation d'une matrice consiste à ajouter un nombre aléatoire à un des nombres de la matrice choisi aléatoirement. La valeur ajoutée est engendrée selon la loi gaussienne d'écart type σ_A ou σ_B selon la matrice considérée. Seules les valeurs positives, engendrées par ces lois, sont acceptées. Les matrices sont alors normalisées de manière à conserver leurs propriétés stochastiques. Les écarts types σ_A et σ_B sont alors ajustés de manière à obtenir un taux d'acceptation de 10%, et de manière à ce que σ_A et σ_B ne dépassent pas 0.5. L'acceptation est régie par la fonction de probabilité de Metropolis. La température initiale est fixée à 1 tandis que la température finale est fixée à 0.000005. La fonction U est d'abord linéaire puis exponentielle. Cette approche, bien qu'intéressante, nécessite de nombreuses itérations (60000 à 400000 itérations pour 64 paramètres selon l'auteur) et n'a fait l'objet d'une comparaison qu'avec l'algorithme de Baum-Welch. L'algorithme de D. B. Paul dépassant légèrement les performances de l'algorithme de Baum-Welch.

2.3.3 Deuxième application du recuit simulé pour l'apprentissage de MMC

L'approche suivie dans (Hamam and Al Ani, 1996) est différente de la précédente. En effet, même si le critère maximisé reste la vraisemblance, les auteurs ont considéré le problème comme un problème discret. Ils utilisent la séquence des états cachés comme espace de recherche. Les auteurs auraient pu utiliser l'algorithme de *segmental k-means* pour apprendre le modèle à partir de la séquence d'états cachés mais leur objectif est de maximiser la vraisemblance et non pas de maximiser la vraisemblance de la séquence d'états cachés la plus probable. Dans leur réalisation, ils cherchent la séquence des états cachés qui, après transformation en MMC par l'algorithme d'apprentissage étiqueté (cf. section 1.6.1), maximise la vraisemblance. Cette approche possède un avantage : l'espace de recherche est fini dénombrable, et un inconvénient : rien ne garantit que cette transformation en MMC parcoure suffisamment l'espace des MMC. Dans (Hamam and Al Ani, 1996), la perturbation d'une séquence d'états cachés est obtenue en choisissant aléatoirement un instant t dans la séquence et en changeant l'état caché au temps t par un autre état caché choisi aléatoirement. L'acceptation est régie par la fonction de probabilité de Metropolis. La baisse de température est géométrique ($U(T) = \beta T$) et par paliers (plusieurs solutions sont testées pour une même température). La longueur des paliers augmente avec la baisse de température. La température initiale est fixée de manière empirique.

2.4 La méthode de recherche tabou

2.4.1 Principe de la recherche tabou

La recherche tabou également appelée recherche avec tabous est une métaheuristique d'optimisation similaire à une descente de gradient (Hertz et al., 1992), mais elle a la capacité de ne pas toujours suivre le sens de la descente. Ses origines datent des années 60, mais ce n'est qu'en 1986 avec les travaux de Fred Glover (Glover, 1986) (Glover and Laguna, 1997) (Glover, 1989a) (Glover, 1989b) que ses principes sont clairement définis. Bien que peu de résultats théoriques sur sa convergence existent (Dréo et al., 2003), la méthode obtient des performances comparables à d'autres grandes métaheuristiques. La recherche tabou repose

principalement sur trois notions (Dréo et al., 2003) : le voisinage, la liste tabou et l'aspiration. L'algorithme de recherche tabou permettant de maximiser la fonction f sur l'espace des solutions $\mathcal{S}$ est donné par l'algorithme 2.4.

Algorithme 2.4: Méthode de recherche tabou

choisir s^* une solution initiale dans $\mathcal{S}$
$s_1 = s^*$, $t = 1$
Répéter
  Construire un voisinage V_{t+1} de la solutions s_t
  Construire l'ensemble des solutions tabous $T_{t+1} \subseteq V_t$
  Construire l'ensemble des solutions « aspirées » $A_{t+1} \subseteq V_t$
  Construire l'ensemble des solutions candidates $C_{t+1} = V_{t+1} - T_{t+1} + A_{t+1}$
  Choisir la meilleure solution de $s_{t+1} \in C_{t+1}$ au sens de f
  Si $f(s_{t+1}) > f(s^*)$ **Alors**
    $s^* = s_{t+1}$
  Fin Si
  Mettre à jour la liste tabou
  $t = t + 1$
Tant que condition d'arrêt non vérifiée

Le fonctionnement de la méthode est simple et est schématisé par la figure 2.2. A partir d'une solution s_t, on construit un ensemble de solutions voisines V_{t+1} à l'aide de transformations simples. V_{t+1} joue un rôle important dans la réussite ou l'échec de la méthode, mais également dans les temps de calculs nécessaires. Pour un bon fonctionnement de la méthode, ce voisinage doit contenir suffisamment de solutions mais, pour des calculs rapides, il doit en contenir le moins possible. Dans la pratique, on ne construit pas l'ensemble complet des solutions voisines de s_t, car il arrive très souvent que le nombre de solutions voisines soit très grand. Bien qu'en théorie, lorsque le voisinage n'est pas complet, rien ne garantit le bon fonctionnement de la méthode, il a été constaté, dans de nombreuses implantations, que les performances de la méthode n'en étaient que peu modifiées (Dréo et al., 2003).

A partir de V_{t+1}, on construit le sous ensemble T_{t+1} de solutions interdites ou tabous. Cet ensemble de solutions interdites a pour objectif d'empêcher la méthode de retourner à des solutions déjà explorées précédemment, ou du moins depuis un certain laps de temps. Cet ensemble de solutions tabous peut être construit de deux façons (Dréo et al., 2003) : soit en mémorisant les solutions effectivement explorées, soit en mémorisant les transformations ayant permis d'arriver à la solution courante. La première possibilité a pour principal intérêt de n'interdire que les solutions effectivement explorées, mais elle possède comme gros inconvénient de nécessiter de conserver de nombreuses solutions dans cette liste. La deuxième possibilité nécessite souvent de mémoriser beaucoup moins d'informations, mais il est possible qu'elle rende tabou des solutions non visitées. En pratique, la deuxième possibilité est utilisée et grâce à l'aspiration (cf. ci-après), les performances de la méthode ne sont pas modifiées. L'interdiction de solutions est provisoire, dans le sens où les interdictions sont oubliées au bout d'un certain temps.

Le deuxième sous-ensemble A_{t+1} construit est celui des solutions, dites « aspirées », vérifiant le critère d'aspiration. Le critère d'aspiration est le moyen de considérer des solutions qui seraient rejetées en raison de leurs qualifications de tabou. Ce critère peut prendre différentes formes. Sa forme la plus simple et la plus souvent utilisée consiste à qualifier d'« aspirée » toute solution meilleure que la meilleure solution déjà trouvée. Cela peut paraître absurde mais, lorsque l'on considère la liste tabou comme une liste de transformations, il arrive fréquemment que des solutions non explorées soient qualifiées de tabou. Il est donc nécessaire de

les rendre possibles, lorsqu'elles permettent d'atteindre une meilleure solution (Hertz et al., 1992).

L'étape suivante consiste alors à choisir la meilleure solution au sens de f de V_{t+1} qui ne soit pas tabou, ou qui soit « aspirée ». Si aucune solution n'est possible (cas $C_{t+1} = \emptyset$), l'algorithme est arrêté, ou redémarré avec une solution initiale aléatoirement choisie. La dernière étape consiste à mettre à jour, si besoin, la meilleure solution connue et la liste tabou.

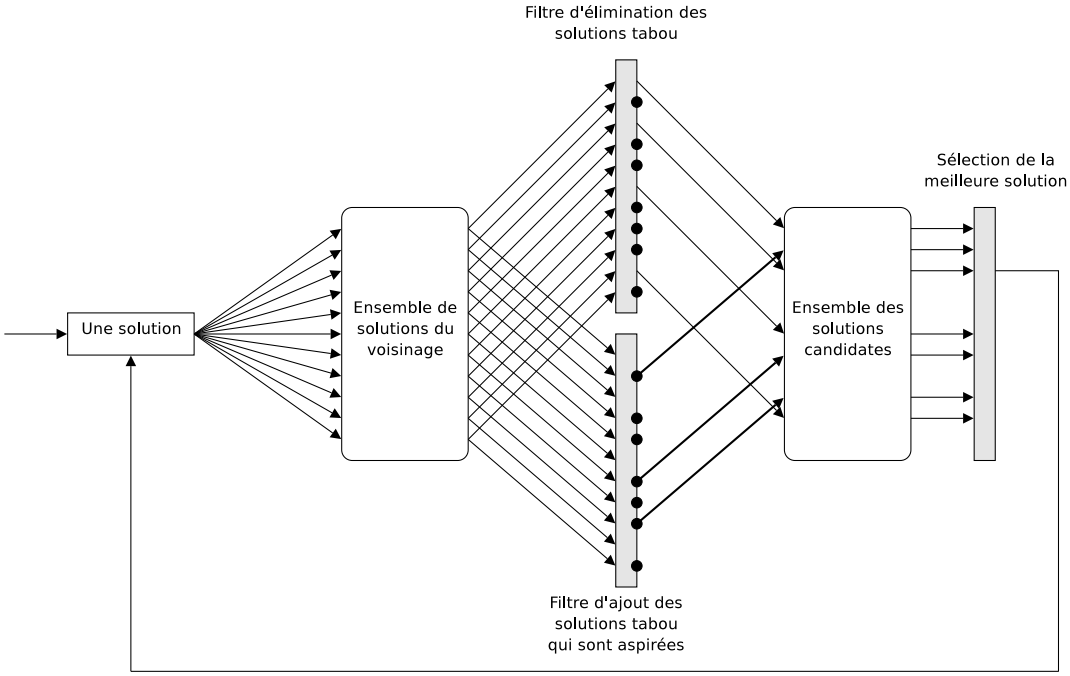


FIG. 2.2 – Principe du calcul des solutions candidates dans la méthode de recherche tabou.

2.4.2 La recherche tabou pour l'apprentissage de MMC

Une adaptation de la recherche tabou à l'apprentissage de MMC a été réalisée récemment dans (Chen et al., 2004). Le critère C considéré dans ces travaux est le critère de maximum de vraisemblance. L'approche utilisée, bien qu'assez naïve, a pour principal avantage d'être rapide. Les solutions manipulées par l'algorithme sont des MMC. La solution initiale est choisie aléatoirement. Le voisinage d'un modèle est obtenu par permutation de deux probabilités « compatibles ». Deux probabilités sont dites « compatibles » si elles font partie de la même contrainte de stochasticité, c'est-à-dire les probabilités de Π ou chacune des lignes des matrices A et B. Le voisinage est considéré de taille fixe. La liste tabou est constituée de la liste des permutations effectuées. Lorsque toutes les solutions sont tabou, un nouveau voisinage est engendré. La méthode est exécutée pour un nombre fixé *a priori* d'itérations. Les auteurs ont montré expérimentalement que cette méthode permettait d'obtenir des résultats légèrement supérieurs à ceux obtenus avec l'algorithme de Baum-Welch sur des MMC gauche-droite (A est triangulaire), mais uniquement sur des modèles ayant une structure très particulière.

2.5 Les algorithmes génétiques

2.5.1 Principe des algorithmes génétiques

Les algorithmes génétiques (AG) proviennent de la modélisation de la théorie de l'évolution de C. Darwin (Darwin, 1859). Les AG font partie du cadre plus général des métaheuristiques

évolutionnaires comprenant, entre autres, les AG (Holland, 1975), les stratégies d'évolution (Beyer, 2001) et la programmation évolutionnaire (Fogel et al., 1966). Initialement développés par J. H. Holland en 1975 (Holland, 1975), les AG sont devenus populaires à partir de la publication du livre « Genetic algorithms in search, optimization and machine learning » de D. E. Goldberg (Goldberg, 1989). La forme canonique d'un algorithme génétique est donnée par l'algorithme 2.5. La figure 2.3 présente son principe sous la forme d'un schéma.

Algorithme 2.5: Algorithme génétique canonique

Initialiser la population d'individus : P_0
Évaluer les individus de la population P_0
$t = 0$
Répéter
  Sélectionner les individus pour la reproduction : $P_t^{\text{parent}} \subseteq P_t$
  Croiser les individus de P_t^{parent} : $P_t^{\text{enfant-1}}$
  Muter les individus de P_t^{enfant} : $\tilde{P}_t^{\text{enfant}}$
  Sélectionner les individus de $P_t^{\text{parent}} \cup \tilde{P}_t^{\text{enfant}}$ à conserver
  Les individus sélectionnés forment la population P_{t+1}
  $t = t + 1$
Tant que condition d'arrêt non vérifiée

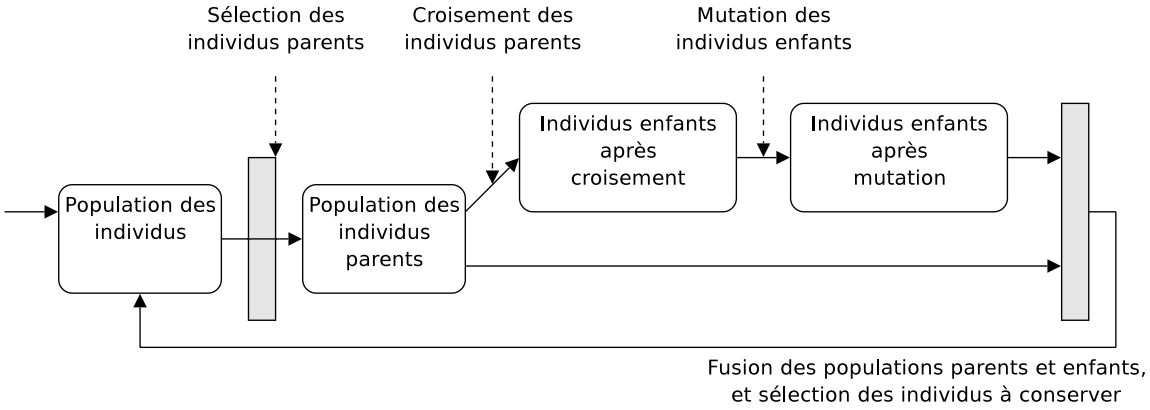


FIG. 2.3 – Principe de l'algorithme génétique canonique.

Un AG (Michalewicz, 1992) consiste, au fur et à mesure des itérations (ou générations), à modifier une population d'individus caractérisés chacun par un chromosome. La modification de la population s'effectue à l'aide de deux types d'opérateurs (Dréo et al., 2003) : les opérateurs de sélection et de variation. Les opérateurs de sélection sont de deux types. Le premier type concerne les opérateurs permettant la sélection des individus à varier. Il inclut entre autres les opérateurs de sélection élitiste (les n meilleurs individus), les opérateurs de sélection proportionnelle à la fonction objectif, la sélection selon le rang et la sélection par tournois. Le deuxième type d'opérateur est l'opérateur de sélection des individus à conserver, également appelé opérateur de sélection pour le remplacement.Il peut également prendre plusieurs formes, dont notamment le remplacement générationnel (on ne conserve que les individus enfants) et le remplacement élitiste (on ne conserve que les meilleurs individus). Les opérateurs de variation sont généralement de deux types : le croisement et la mutation. Le croisement consiste à fabriquer un nouvel individu qui « hérite » en partie des caractéristiques de ses deux parents. La mutation consiste à introduire une part de modification aléatoire dans les individus, afin d'assurer une certaine variété. Les opérateurs de variation dépendent fortement

du codage des individus.

Dans la littérature, on peut trouver principalement trois utilisations des algorihmes génétiques pour l'apprentissage des MMC. La première approche cherche un modèle dont le nombre d'états cachés est fixé par un algorithme génétique parallèle. Les deux approches suivantes cherchent toutes les deux à trouver à la fois le nombre d'états cachés et le bon paramétrage du meilleur modèle. Ces trois approches sont décrites ci-dessous.

2.5.2 Apprentissage par un algorithme génétique parallèle

L'apprentissage d'un ensemble de séquences d'observations $\{O_1, \ldots, O_K\}$ par un algorithme parallèle a été réalisé par Kwong et Chau (Kwong and Chau, 1997). Le critère optimisé est la moyenne des vraisemblances des séquences d'observations par rapport au modèle $\frac{1}{K}\sum_{k=1}^{K} P(V = O_k/\lambda)$. Dans cet AG, les individus sont représentés par les coefficients des matrices A et B linéarisés et mis bout à bout. La sélection employée est une sélection élitiste. Le croisement de deux individus s'effectue en probabilité. Lorsque le croisement n'est pas appliqué, les deux individus parents prennent la place des deux individus enfants. Lorsque le croisement est effectué, les individus enfants sont obtenus par l'opérateur de croisement à un point (1X) sur la matrice A et par l'opérateur de croisement à deux points (2X) sur la matrice B. La mutation s'effectue aussi en probabilité. Elle consiste à modifier aléatoirement un coefficient de A et deux coefficients de B. Les nouveaux coefficients sont choisis aléatoirement. Finalement, les matrices sont normalisées afin de respecter les contraintes de stochasticité.

Pour réduire les temps d'exécution, les auteurs ont utilisé le modèle *island* pour paralléliser les calculs. Dans le modèle *island* (Michalewicz, 1992), plusieurs populations évoluent en parallèle et des opérateurs de migration permettent l'échange et/ou le croisement non systématique d'individus de deux populations. Dans cette adaptation, chaque noeud du système parallèle transmet, à la fin de chaque itération, les 10 meilleurs individus de sa population dans une mémoire partagée et remplace les 10 plus mauvais individus par les 10 meilleurs de la mémoire partagée.

Dans les expérimentations menées par les auteurs, il apparaît que les performances obtenues sont significativement meilleures que celles obtenues par l'algorithme de Baum-Welch ; toutefois, il est important de noter que les expérimentations ont été effectuées sur un MMC gauche-droite possédant 5 états cachés.

2.5.3 Apprentissage de la topologie

Dans (Thomsen, 2002), l'auteur effectue l'apprentissage de MMC à l'aide d'un algorithme génétique, avec l'objectif de se servir du modèle obtenu avec l'algorithme de Viterbi, afin de segmenter des séquences de protéines en trois catégories d'états : α, β et autre. Plusieurs séquences d'observations discrètes $O_1, \ldots, O_L$ sont apprises. Les individus manipulés par l'algorithme sont constitués des trois matrices stochastiques du modèle correspondant. Puisque l'objectif est de segmenter en trois parties, un minimum de trois états cachés est imposé. Les individus de la population initiale ont entre 3 et 10 états cachés dans les expérimentations menées dans (Thomsen, 2002). L'opérateur de variation consiste à appliquer l'opérateur de croisement, suivi de l'opérateur de mutation.

L'opérateur de croisement ne s'applique qu'aux matrices B des individus. Les croisements considérés sont le croisement uniforme, le croisement arithmétique, le croisement à 1 point et le croisement à 2 points. Le type de croisement utilisé est choisi de manière aléatoire et uniforme.

Les opérateurs de mutation sont :
- l'ajout d'un état caché (les nouvelles probabilités sont aléatoires, la matrice A est normée),
- la suppression d'un état caché (la matrice A est normée),
- l'ajout d'une transition (un coefficient $a_{i,j}$ nul est rendu non nul),
- la suppression d'une transition (un coefficient $a_{i,j}$ non nul est rendu nul),
- l'échange de deux probabilités (une probabilité $a_{i,j}$ est échangée avec la probabilité $a_{i,k}$),
- la modification d'une probabilité selon une loi gaussienne de variance $1/(t+1)$.

Le type de mutation effectuée est choisi aléatoirement et uniformément. Le croisement et la mutation sont appliqués avec des probabilités définies *a priori*. Un individu enfant remplace l'individu parent si la valeur de la fonction objectif est supérieure à celle de l'individu parent.

La fonction objectif utilise la vraisemblance et le critère d'information bayesienne (*Bayesian Information Criterion*). Sa valeur pour l'individu λ est donnée par :

$$\sum_{l=1}^{L} \ln P(V = O_l/\lambda) + \omega \frac{\ln(L+1)}{2} N(N+M+1)$$

Le coefficient ω permet de contrôler l'importance de la vraisemblance par rapport à la complexité du modèle. Finalement, la meilleure solution produite est soumise à l'algorithme de Baum-Welch pour l'ajustement final. Les auteurs ont uniquement montré dans leurs expériences que le MMC obtenu était meilleur que le MMC obtenu par l'algorithme de Baum-Welch.

2.5.4 L'algorithme GHOSP

Dans (Slimane et al., 1999) et (Brouard, 1999), les auteurs ont adopté une approche différente. L'algorithme GHOSP (*Genetic hybridation optimization and search of parameters*) est un algorithme génétique qui fonctionne en coopération avec l'algorithme de Baum-Welch : on parle d'hybridation. Dans cet algorithme, les individus ont pour chromosome la matrice $C = (\Pi, A, B)$ de dimension $N \times N + M + 1$ obtenue par concaténation des matrices Π, A et B du modèle. La population est constituée d'individus ayant un nombre d'états cachés différents. A l'initialisation, N_c MMC, dont le nombre d'états cachés est fixé entre $N_{\min}$ et $N_{\max}$, sont engendrés aléatoirement. La population est donc constituée de $N_c * (N_{\max} - N_{\min} + 1)$ MMC. La fonction objectif à maximiser est la vraisemblance.

La sélection des individus parents est une sélection élitiste des N_{parents} meilleurs individus. L'opérateur de variation se décompose en trois opérateurs : l'opérateur de croisement, l'opérateur de mutation et l'opérateur d'optimisation locale. L'opérateur de croisement considère deux MMC de même nombre d'états cachés choisis aléatoirement dans la population parente.

L'opérateur de croisement est un opérateur de croisement à un point (1X). Un point de coupure aléatoire est choisi dans l'intervalle $[1; N]$ en désignant par N, le nombre d'états cachés des individus. Les chromosomes sont coupés horizontalement à ce point de coupure et les deux parties basses sont échangées. La stochasticité des matrices A et B est conservée mais pas celle des matrices Π. Il est donc nécessaire de les normaliser. Un seul individu enfant est ajouté dans la population des enfants parmi les deux MMC obtenus. L'opérateur est appliqué jusqu'à ce que la somme des tailles de la population parente et enfant corresponde à la taille de la population totale.

L'opérateur de mutation s'applique sur les individus enfants retenus. Chaque coefficient du chromosome est modifié aléatoirement selon la probabilité P_{mut}. La mutation d'un coefficient consiste à définir aléatoirement et uniformément sa valeur dans $[0; 1]$. L'opérateur de mutation se termine par la normalisation des matrices, afin de conserver la stochasticité. L'opérateur d'optimisation locale consiste à appliquer l'algorithme de Baum-Welch à tous les individus de

la population. L'algorithme n'est pas exécuté jusqu'à convergence, mais uniquement pendant un certain nombre d'itérations fixé *a priori*. Les auteurs ont montré que leur algorithme donnait des MMC significativement meilleurs que l'algorithme de Baum-Welch.

D'autres variantes d'algorithmes génétiques pour l'apprentissage de MMC ont été créées dans la thèse de T. Brouard (Brouard, 1999) mais, d'après l'auteur, ils obtiennent des performances inférieures à celles de l'algorithme GHOSP.

2.6 Apprentissage incrémental à base de population

2.6.1 Principe de l'apprentissage incrémental à base de population

L'apprentissage incrémental à base de population (*population-based incremental learning* PBIL) (Bulaja and Caruana, 1995) (Bulaja, 1994) est une métaheuristique cousine des algorithmes génétiques. Conformément à la description donnée dans (Dréo et al., 2003), l'apprentissage incrémental à base de population peut être considéré comme un cas particulier des algorithmes à estimation de distribution, à la différence que l'estimation des probabilités s'effectue, dans le cas du PBIL, sous la forme d'un renforcement. PBIL ne considère les individus que sous la forme de chaînes binaires, tout comme le fait la forme la plus simple de l'AG. La principale différence consiste dans la gestion de la population : PBIL ne maintient pas une population. En lieu et place, il conserve un vecteur de probabilité décrivant, de manière statistique, la présence ou l'absence de 1 aux différentes positions des chaînes binaires. L'ajustement du vecteur de probabilité suit une règle similaire à l'apprentissage par renforcement. L'algorithme PBIL est donné par l'algorithme 2.6 et la figure 2.4 schématise son principe.

Algorithme 2.6: Algorithme d'apprentissage incrémental à base de population

Pour $i = 1$ **à** L **Faire**
  $P_i = 0.5$
Fin Pour
Répéter
  Pour $k = 1$ **à** K **Faire**
    Engendrer aléatoirement un vecteur binaire v_k selon les probabilités $(P_i)_i$
    Evaluer la vecteur v_k
  Fin Pour
  Trier les vecteurs du meilleur au plus mauvais
  Pour $u = 1$ **à** U **Faire**
    Pour $i = 1$ **à** L **Faire**
      $P_i = P_i * (1 - \tau) + v_u[i] * \tau$
    Fin Pour
  Fin Pour
Tant que condition d'arrêt non vérifiée

Initialement, les L probabilités d'apparition d'un 1 dans les chaînes sont fixées à 0.5. Lors de chaque itération de l'algorithme, K vecteurs binaires (individus) de longueur L sont engendrés aléatoirement suivant les probabilités de présence de 1 à chacune des positions. Ces K individus sont évalués et triés en ordre décroissant d'intérêt. Les U meilleurs individus sont utilisés pour mettre à jour les probabilités, suivant une règle d'apprentissage par renforcement.

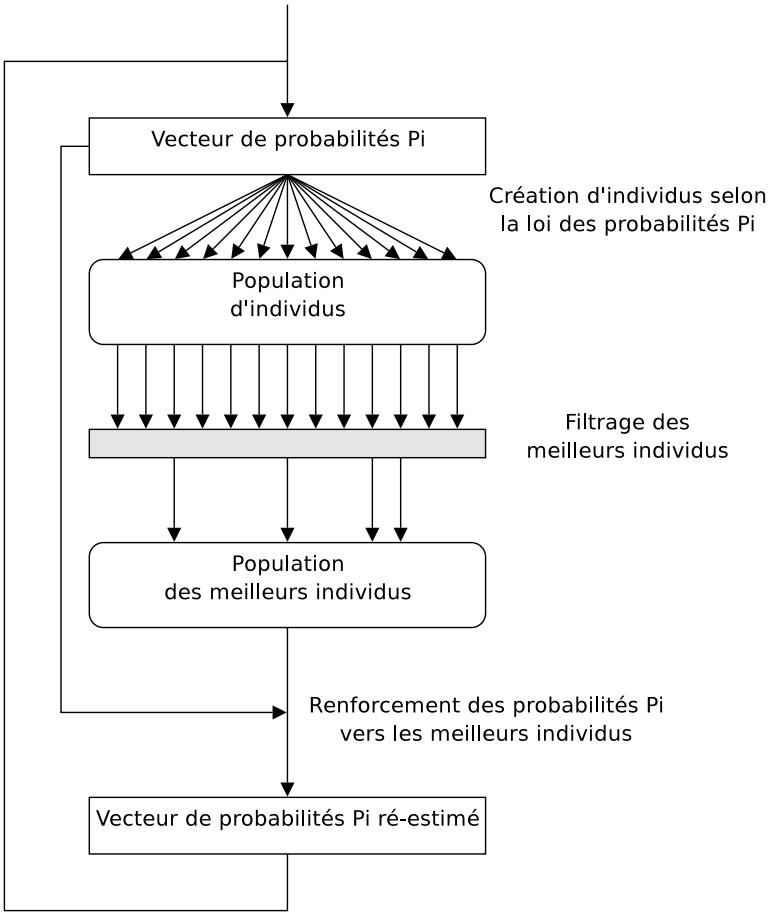


FIG. 2.4 – Principe de l'apprentissage incrémental à base de population.

2.6.2 Apprentissage de MMC par l'algorithme d'apprentissage incrémental à base de population

Le PBIL a été utilisé dans (Maxwell and Anderson, 1999) pour l'apprentissage de MMC. Dans l'implantation réalisée, les auteurs ont introduit, comme suggéré dans (Bulaja and Caruana, 1995), une phase de mutation du vecteur de probabilités afin d'éviter de converger trop vite vers des optima locaux. A cette fin, chaque probabilité P_i subit une mutation, avec une probabilité fixée *a priori*, selon la formule $P_i = P_i * (1 - \rho) + \rho * v_u[i]$. Le paramètre ρ contrôle la quantité de mutation que subit P_i vers la valeur du vecteur du meilleur individu obtenu à l'itération courante. La mise à jour du vecteur de probabilité ne dépend que du meilleur individu obtenu *i.e.* $U = 1$. Pour coder les MMC en chaîne binaire, les auteurs de (Maxwell and Anderson, 1999) ont choisi de discrétiser les probabilités et d'adopter un codage qui incorpore naturellement les contraintes de stochasticité et évite les « sauts » de valeurs trop brusques (ex : le passage de 0000 à 1000 fait passer la valeur 0 à une valeur de 1 en changeant seulement un bit). Pour décrire ce codage, considérons le vecteur stochastique $\Pi = (\pi_1, \ldots, \pi_N)$. Pour éliminer la contrainte stochastique, les auteurs ont transformé Π en Π' grâce à la formule $\pi_i = \pi'_i(1 - \sum_{1 \leq k < i} \pi_k)$, c'est-à-dire $\pi'_i = \frac{\pi_i}{1 - \sum_{1 \leq k < i} \pi_k}$. Ce codage est intéressant car les π'_i prennent leur valeur dans l'intervalle $[0; 1]$, qui est alors discrétisé sur 16 bits afin d'obtenir la représentation binaire de chacun des nombres de $[0; 1]$. Cet algorithme a été utilisé pour effectuer l'apprentissage avec le critère de vraisemblance et de maximisation de l'information mutuelle avec des résultats mitigés, mais semblant donner l'algorithme de Baum-Welch vainqueur pour le premier critère.

2.7 La métaheuristique de colonie de fourmis API

2.7.1 Principe de la métaheuristique de colonie de fourmis API

API est une métaheuristique d'optimisation élaborée par N. Monmarché au cours de sa thèse (Monmarché, 2000) (Monmarché, 2002). Cette métaheuristique est inspirée du comportement de prédation d'une population de fourmis primitives : *Pachycondyla apicalis*.

Les espèces de fourmis les plus communes, qui ont inspiré de nombreuses métaheuristiques d'optimisation, telles que ACO, utilisent la *stigmergie*[2] à travers le dépôt par les fourmis dans leur environnement de substances chimiques appelées phéromones. Cependant, il existe des espèces n'utilisant pas de phéromones : les *Pachycondyla apicalis* en sont un exemple. Nous n'allons pas décrire en détail les caractéristiques des fourmis *Pachycondyla apicalis*, car cela nous emmènerait trop loin de notre propos. De manière à introduire la métaheuristique API, il est néanmoins nécessaire de décrire les principes généraux du fourragement (recherche de nourriture) de ces fourmis.

Les caractéristiques des fourmis *Pachycondyla apicalis*, qui ont conduit à la création de la métaheuristique API, sont (Monmarché et al., 2000) :

- les fourmis vivent en petite colonie ;
- comme elle ne sait pas construire de fourmilière, la colonie installe son nid dans des souches de bois en décomposition, ce qui l'oblige à le déménager régulièrement ;
- elles n'utilisent pas de phéromones, donc elles n'utilisent pas de mécanisme de recrutement de masse ;
- elles sont capables de mémoriser visuellement un chemin et donc de revenir à un endroit qu'elles ont mémorisé ;
- chaque fourmi sort du nid pour aller chasser et revient au bout d'un certain temps au nid ;
- chaque fourmi possède son (ses) site(s) de chasse ;
- si la recherche précédente a été fructueuse, alors elle retourne sur le même site de chasse puis reprend sa recherche d'une proie dans la zone. Dans le cas contraire, au bout d'un certain nombre d'échecs, la fourmi abandonne le site. Si la fourmi ne connaît plus (ou pas encore) de site de chasse, alors elle sort du nid dans une direction aléatoire et choisit un site aléatoirement.

Ces propriétés sont donc particulièrement intéressantes car elles permettent à partir d'un comportement simple d'obtenir une bonne couverture de l'espace de recherche.

L'adaptation du comportement de fouragement des fourmis *Pachycondyla apicalis* au problème de la maximisation d'une fonction a donné naissance à la métaheuristique API, en considérant simplement que la capture d'une proie correspond à l'amélioration de la fonction objectif. On considère une fonction réelle f à maximiser définie sur $\mathcal{S}$. La métaheuristique API s'appuie principalement sur deux opérateurs $\mathcal{O}_{\text{Rand}}$ et $\mathcal{O}_{\text{Explo}}$ spécifiques à l'espace $\mathcal{S}$. Ces deux opérateurs sont définis par :

- $\mathcal{O}_{\text{Rand}}(\mathcal{S})$: l'opérateur permettant d'obtenir aléatoirement une solution dans l'espace de recherche $\mathcal{S}$.
- $\mathcal{O}_{\text{Explo}}(s)$: l'opérateur permettant d'obtenir aléatoirement une solution dans le « voisinage » de la solution s. Cet opérateur est utilisé deux fois par API : pour créer un site de chasse et pour explorer une solution autour d'un site de chasse. Pour créer un site de chasse (cf. figure 2.5), on applique l'opérateur $\mathcal{O}_{\text{Explo}}$ à la position du nid. Dans la suite, nous notons cet opérateur : $\mathcal{O}_{\text{ExploSite}}$. Pour explorer une solution autour d'un site de chasse (cf. figure 2.6), on applique l'opérateur $\mathcal{O}_{\text{Explo}}$ à la position du site de chasse. Dans la suite, nous notons cet opérateur : $\mathcal{O}_{\text{ExploLocal}}$.

[2] On parle de *stigmergie* lorsque deux individus utilisent leur environnement pour échanger des informations (Monmarché, 2000).

La métaheuristique API est présentée par l'algorithme 2.7. Le lecteur notera, cependant, qu'une version plus détaillée de cet algorithme est donnée en annexe C. Les paramètres de API sont :

- $\mathcal{T}_{\max}$: le nombre d'itérations de l'algorithme. Cette valeur correspond au nombre de sorties du nid effectuées par chacune des fourmis ;
- $\mathcal{T}_{\text{Déplacement}}$: le nombre d'itérations de l'algorithme entre deux déplacements du nid vers la meilleure solution obtenue (cf. figure 2.7) ;
- $\mathcal{N}$: le nombre de fourmis ;
- $\mathcal{M}_{\max}(a_i)$: la taille de la mémoire de la fourmi a_i *i.e.* le nombre de sites de chasse de la fourmi ;
- $e_{\max}(a_i)$: la patience de la fourmi a_i *i.e.* le nombre maximum d'essais successifs infructueux autour d'un site avant d'oublier un site de chasse.

Bien entendu, les opérateurs $\mathcal{O}_{\text{Rand}}$ et $\mathcal{O}_{\text{Explo}}$ peuvent possèder des paramètres particuliers, mais ceux-ci dépendent du problème considéré.

Algorithme 2.7: La métaheuristique API

$s_{\text{Nid}} = \mathcal{O}_{\text{Rand}}$
La mémoire des fourmis est initialisée à vide
Pour $t = 1$ **à** $\mathcal{T}_{\max}$ **Faire**
  Pour chacune des fourmis **Faire**
    Si la fourmi n'a pas encore choisi tous ses sites de chasse **Alors**
      La fourmi crée un nouveau site de chasse (cf. figure 2.5)
    Sinon
      Si la dernière solution explorée est un échec **Alors**
        Choisir aléatoirement un nouveau site de chasse à explorer
      Fin Si
      Explorer une solution autour du site de chasse à explorer (cf. figure 2.6)
      Si la nouvelle solution est meilleure que le site de chasse **Alors**
        Remplacer le site de chasse
      Sinon
        Incrémenter le compteur d'échec
        Si trop d'échec **Alors**
          Oublier le site de chasse
        Fin Si
      Fin Si
    Fin Si
  Fin Pour
  Si t modulo $\mathcal{T}_{\text{Déplacement}} = 0$ **Alors**
    Déplacer le nid sur la meilleure solution trouvée (cf. figure 2.7)
    Vider la mémoire des fourmis
  Fin Si
Fin Pour

2.7.2 Apprentissage de MMC par l'algorithme API

L'application de l'algorithme API à l'apprentissage de MMC pour le critère de maximum de vraisemblance a été réalisé par N. Monmarché dans (Monmarché, 2000). Dans cette application, l'espace de recherche est composé des matrices stochastiques A, B et Π décrivant un MMC.

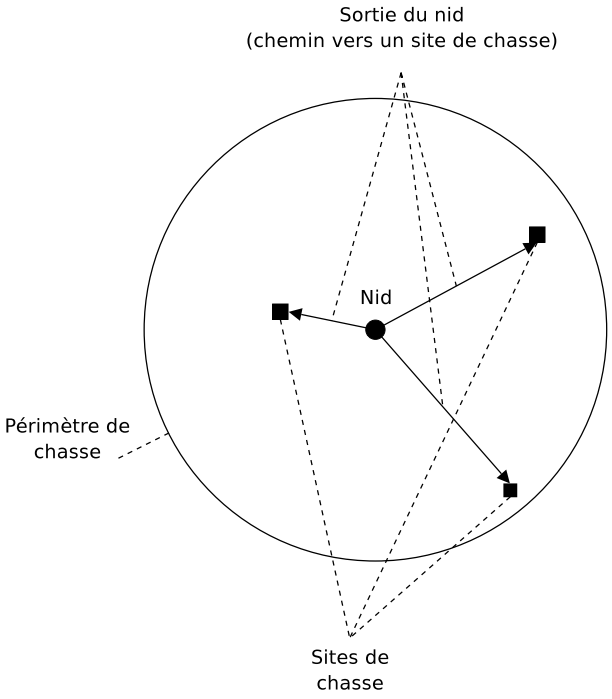


FIG. 2.5 – La fourmi sort du nid et choisit trois sites de chasse dans le périmètre de chasse (défini autour du nid).

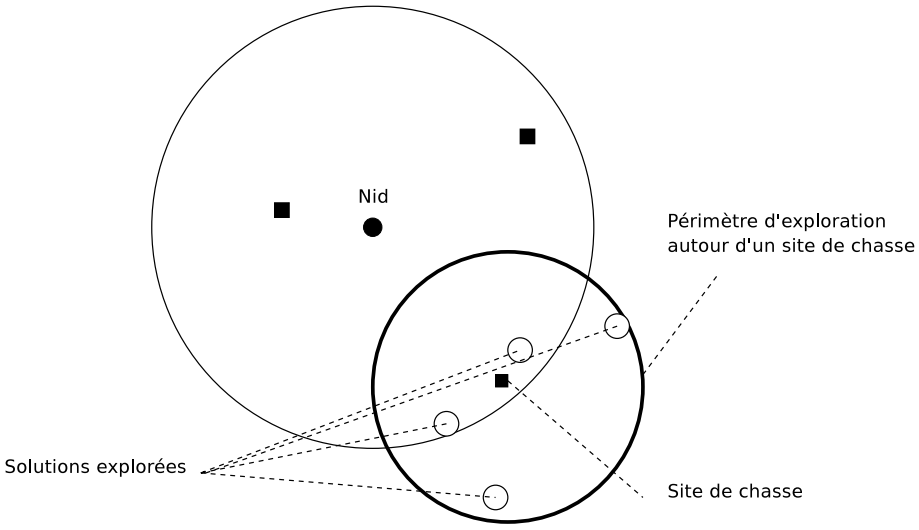


FIG. 2.6 – La fourmi choisit un site de chasse et explore 4 solutions dans le périmètre du site de chasse.

Les opérateurs qui ont été utilisés sont :

- $\mathcal{O}_{\text{Rand}}(\mathcal{S})$, l'opérateur permettant de créer aléatoirement un MMC, consiste à engendrer aléatoirement des matrices à coefficients positifs et à normer la somme des lignes à 1 de manière à obtenir des matrices stochastiques.
- L'opérateur $\mathcal{O}_{\text{Explo}}$ est paramétré par une amplitude A. Nous adopterons donc les notations $\mathcal{O}_{\text{ExploSite}}(s, \mathcal{A}_i^{\text{Nid}})$ et $\mathcal{O}_{\text{ExploLocal}}(s, \mathcal{A}_i^{\text{Local}})$ pour les opérateurs en désignant par $\mathcal{A}_i^{\text{Nid}}$ l'amplitude pour la création des sites de chasse pour la fourmi a_i et $\mathcal{A}_i^{\text{Local}}$ l'amplitude pour l'exploration des sites de chasse par la fourmi a_i. L'opérateur d'exploration consiste à appliquer la fonction $AM_{\mathcal{A}}$ (dépendante de l'amplitude $\mathcal{A}$) à chacun des co-

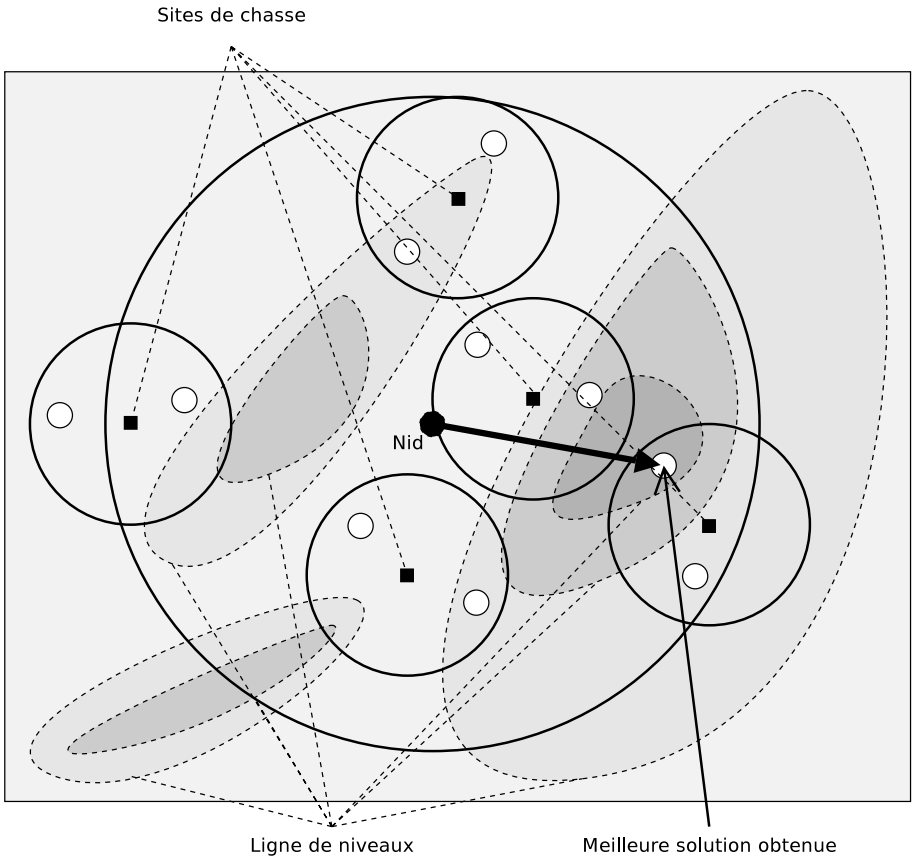


FIG. 2.7 – Déplacement du nid dans l'algorithme API sur la meilleure solution obtenue.

efficients des matrices, puis à normaliser les sommes à 1, de manière à préserver la stochasticité des matrices. Le modèle ainsi obtenu est alors soumis à l'algorithme de Baum-Welch, afin de subir une optimisation locale. La valeur de la fonction $AM_{\mathcal{A}}$ en x est définie par :

- $v = x + \mathcal{A} \cdot (2\mathcal{U}([0,1[) - 1)$
- $AM_{\mathcal{A}}(x) = \begin{cases} -v & \text{si } v < 0 \\ 2 - v & \text{si } v > 1 \\ v & \text{sinon} \end{cases}$

Des paramétrages homogènes et hétérogènes des amplitudes ont été appliqués dans (Monmarché, 2000). L'auteur a montré sur un exemple très réduit d'observations construites artificiellement que cet algorithme donnait de meilleurs résultats que les algorithmes *Random+optim* (cf. section 2.2) et GHOSP (cf. section 2.5.4) mais aucun test plus complet n'a été réalisé.

2.8 L'optimisation par essaim particulaire (OEP)

2.8.1 Principe de l'optimisation par essaim particulaire

L'optimisation par essaim particulaire (OEP) (Clerc, 2005a) est une métaheuristique à base de population inspirée du comportement sociologique des nuées d'oiseaux ou de bancs de poissons. Elle a été introduite par Kennedy et Eberhart (Kennedy and Eberhart, 1995). Malgré sa relative jeunesse, cette technique a déjà fait l'objet de nombreuses recherches et de nombreuses adaptations à des problèmes divers se sont révélées fructueuses. (Abido, 2002), (Omran et al., 2002), (Hu et al., 2003), (van der Merwe and Engelbrecht, 2003), (Paquet

and Engelbrecht, 2003), (Salman et al., 2003) et (Tasgetiren et al., 2004) en sont quelques exemples.

L'OEP est une technique simple consistant à faire se déplacer $\mathcal{N}$ particules dans un espace de solutions. Chaque particule possède, à l'instant t, une position notée $x_i(t)$ et une vitesse de déplacement $v_i(t)$.

On note f la fonction objectif, $x_i^+(t)$ la meilleure position trouvée par la particule i à l'instant t, $V_i(t)$ l'ensemble des particules situées dans le voisinage de la particule i et $\hat{x}_i(t)$ la position de la meilleure (au sens de f) particule de $V_i(t)$, *i.e.* $\hat{x}_i(t) = \arg\max_{x_j \in V_i(t)} f(\hat{x_j}(t-1))$ dans le cas de la maximisation de f.

L'OEP classique est contrôlée par trois coefficients : ω, c_1 et c_2, à travers les équations de déplacement des particules :

$$x_i(t+1) = x_i(t) + v_i(t)$$

$$v_i(t+1) = \omega \cdot v_i(t) + c_1 \cdot \mathcal{U}([0,1])(x_i^+(t) - x_i(t)) + c_2 \cdot \mathcal{U}([0,1])(\hat{x}_i(t) - x_i(t))$$

en désignant par $\mathcal{U}([0,1])$ un nombre aléatoire engendré uniformément dans $[0;1]$ (cf. figure 2.8). Dans certaines définitions de l'OEP, les équations suivantes sont utilisées en lieu et place des précédentes :

$$x_i(t+1)_j = x_i(t)_j + v_i(t)_j$$

$$v_i(t+1)_j = \omega \cdot v_i(t)_j + c_1 \cdot \mathcal{U}([0,1])(x_i^+(t)_j - x_i(t)_j) + c_2 \cdot \mathcal{U}([0,1])(\hat{x}_i(t)_j - x_i(t)_j)$$

avec $x_i(t)_j$ représentant la j-ième composante du vecteur $x_i(t)$.

La différence avec les équations précédentes réside dans le mode de mise à jour de la vitesse et de la position : dans un cas, elle s'effectue vectoriellement, dans l'autre, elle s'effectue composante par composante. Dans le premier cas, deux coefficients aléatoires sont nécessaires, tandis que, dans le second, deux coefficients aléatoires sont nécessaires par composante. L'algorithme résultant des premières équations est donné par l'algorithme 2.8. La figure 2.8 illustre le principe de mise à jour de la vitesse et de la position d'une particule.

Le coefficient ω est un coefficient d'inertie contrôlant l'influence de la vitesse actuelle de la particule sur la vitesse à l'instant suivant. Les coefficients c_1 et c_2 contrôlent respectivement l'influence de la meilleure position connue de la particule et de la meilleure position connue des particules du voisinage.

En fonction des coefficients ω, c_1 et c_2 et de la nature du voisinage des particules, l'OEP peut être déclinée en plusieurs catégories :

- Si $V_i(t)$ est l'ensemble des particules, alors on parle d'OEP *gbest* (*global best*) et d'OEP *lbest* (*local best*) sinon ;
- Si $c_1 = 0$, alors la composante cognitive n'intervient pas. On dit que le modèle est social ;
- Si $c_2 = 0$, alors la composante sociale n'intervient pas. On dit que le modèle est cognitif.

La notion de voisinage peut prendre plusieurs formes : sociologique et spatiale. Dans un voisinage sociologique, les particules sont organisées selon un réseau de communication fixe (ex : en étoile, en cercle). Les particules faisant partie du voisinage sont alors celles qui sont reliées par cette structure. Dans un voisinage spatial, le voisinage des particules est calculé à chaque itération de l'algorithme. Il correspond soit aux $|V|$ particules les plus proches, soit aux particules situées dans un certain rayon ($d(x_i(t), x_j(t)) < d_{\max}$).

De nombreuses variantes de l'algorithme initial ont vu le jour, certaines définissent des bornes minimales et maximales de l'amplitude de la vitesse, afin d'éviter une divergence de celle-ci. D'autres utilisent des coefficients variables dans le temps, comme par exemple le coefficient d'inertie ($\lim_{t \to +\infty} \omega(t) = 0$) (Ratnaweera et al., 2004) ou des coefficients de constriction (Clerc, 2005a).

Algorithme 2.8: L'optimisation par essaim particulaire

Initialiser la position $(x_i(0))$ et la vitesse $(v_i(0))$ des particules
Pour $i = 1$ **à** $\mathcal{N}$ **Faire**
  $x_i^+(0) = x_i(0)$
Fin Pour
Pour $t = 1$ **à** $\mathcal{T}_{\max}$ **Faire**
  Pour $i = 1$ **à** $\mathcal{N}$ **Faire**
    $x_i(t+1) = x_i(t) + v_i(t)$
    $v_i(t+1) = \omega \cdot v_i(t) + c_1 \cdot \mathcal{U}([0,1])(x_i^+(t) - x_i(t)) + c_2 \cdot \mathcal{U}([0,1])(\hat{x}_i(t) - x_i(t))$
  Fin Pour
Fin Pour

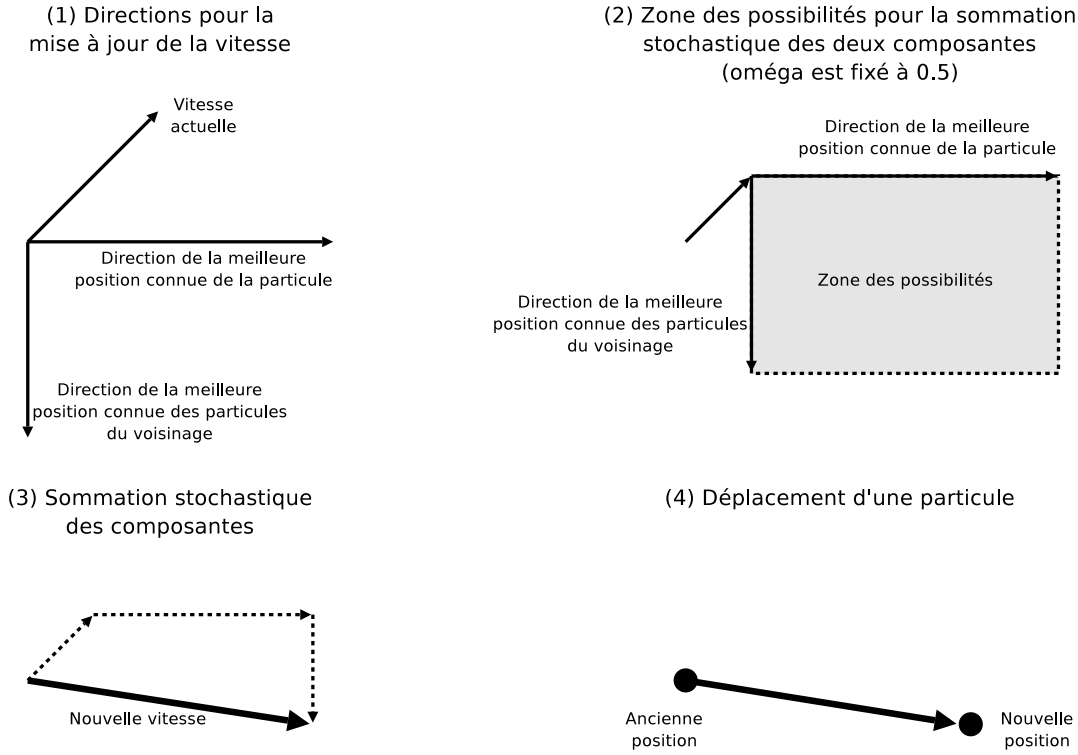


FIG. 2.8 – Principe de mise à jour de la vitesse et de déplacement d'une particule.

2.8.2 Apprentissage de MMC par l'algorithme OEP

L'optimisation par essaim particulaire a été appliquée à l'apprentissage de MMC par Rasmussen et Krink dans (Rasmussen and T. Krink, 2003), avec le critère de maximum de vraisemblance de plusieurs séquences d'observations simultanément ou avec un critère d'alignement décrit dans (Rasmussen and T. Krink, 2003). Les MMC considérés comportent des états d'insertion et de suppression de symboles (Baldi and Brunak, 1998) et une structure très contrainte. Ces MMC possèdent $N + M$ coefficients non nuls. Les particules évoluent dans l'espace $\mathbb{R}^{N+M}$: les positions et les vitesses sont donc des vecteurs réels en $N + M$ dimensions. Pour évaluer une solution, une copie de la position est effectuée et les coefficients sont normalisés, de manière à satisfaire les contraintes de stochasticité. Le modèle obtenu est alors évalué. Cette évaluation donne l'évaluation de la position avant copie. La population initiale des particules est choisie aléatoirement, à l'exception de deux éléments : la meilleure solution obtenue grâce à un algorithme de recuit simulé et la meilleure solution obtenue

par l'algorithme de Baum-Welch jusqu'à convergence à partir d'une solution aléatoire. Cette approche garantit des performances de l'OEP au moins aussi bonnes que ces deux algorithmes. L'OEP utilisée a été étendue de manière à utiliser une phase de croisement similaire à un algorithme génétique après la mise à jour des positions et des vitesses des particules. Le croisement des positions est effectué à l'aide d'un croisement arithmétique classique, tandis que les vitesses sont remplacées par la moyenne des vitesses des deux particules croisées. Le croisement des particules s'applique avec une probabilité fixée *a priori*. Les auteurs ont montré expérimentalement que cette métaheuristique donnait de bonnes performances pour la segmentation de protéines.

2.9 Conclusion

Comme nous venons de le voir, de nombreuses métaheuristiques ont été utilisées pour l'apprentissage de MMC. Cependant, les travaux évoqués précédemment ne sont pas d'une très grande aide pour l'utilisateur souhaitant apprendre des MMC pour un critère fixé.

En effet, plusieurs problèmes se posent du fait de la non standardisation des expérimentations en la matière. Bien que des études soient menées pour la recherche des modèles optimaux, il a été montré dans de nombreuses expérimentations que des modèles sous-optimaux obtenus par des algorithmes de gradients, ou assimilés, donnaient des résultats satisfaisants. Par conséquent, bien que permettant potentiellement d'améliorer les performances de l'application, la recherche des modèles optimaux n'a pas été jusque là une préoccupation importante. Cette absence d'intérêt n'a pas encouragé la mise en place de jeux de données standardisés permettant la comparaison des différents algorithmes les uns par rapport aux autres.

De plus, les publications en la matière restent souvent peu détaillées, voire confuses, laissant une grande part à l'interprétation pour l'implémentation. Par conséquent, l'implémentation de certaines méthodes en vue de leurs comparaisons reste une pratique coûteuse en temps, empêchant dans la plupart des cas une comparaison fiable.

Finalement, même si les jeux de données étaient standardisés et les algorithmes bien décrits, nous serions encore confrontés au problème du paramétrage des algorithmes. Dans la majorité des travaux évoqués précédemment, le paramétrage des algorithmes correspond à un choix par défaut, plutôt qu'une recherche des paramètres optimaux ou de ceux qui garantissent un bon résultat.

Suite à ces constatations, nous nous sommes attachés dans le chapitre 4 à déterminer les paramètres optimaux des algorithmes que nous proposons et à comparer nos algorithmes avec deux métaheuristiques : l'algorithme génétique GHOSP (cf. section 2.5.4) et l'algorithme de fourmis artificielles API (cf. section 2.7.2). Nous avons considéré ces algorithmes car nous avons accès à leurs implémentations.

Chapitre 3

Les techniques de visualisation de données

De manière à introduire les notions nécessaires au chapitre 6 concernant la visualisation de dissimilarité pour des MMC, nous décrivons dans ce chapitre en quoi consiste la visualisation d'information.

3.1 Introduction

Depuis des siècles, l'être humain cherche à comprendre le monde qui l'entoure. Cette compréhension se traduit principalement par la définition de règles et de lois nous permettant de mieux comprendre notre environnement, de mieux prédire l'avenir et de pouvoir confirmer les hypothèses établies sur ce monde.

Dans le cas de la fouille de données, l'être humain ne cherche plus à comprendre un environnement global, mais plutôt à comprendre une partie de cet environnement. A cet effet, il est à la recherche de structures, de caractéristiques, de motifs, de tendances, d'anomalies ou, de manière plus générale, de la présence ou de l'absence de relations dans les données (Fayyad et al., 2001).

Depuis plus d'un demi-siècle, cette analyse manuelle des données, effectuée dans le but d'en extraire des connaissances, n'est plus réalisable à la main. En effet, du fait de la multiplication des paramètres pris en compte et de la multiplication des mesures effectuées, la quantité d'information à traiter a rendu obligatoire l'utilisation de méthodes pour les analyser.

Le traitement purement automatique de ces informations afin d'en extraire des connaissances se nomme la « fouille de données » ou *data mining*. Bien que permettant une automatisation de l'analyse, la fouille de données n'est pas la solution parfaite au problème de la compréhension des données. En effet, très souvent, les algorithmes de fouilles de données, aussi sophistiqués qu'ils soient, ne parviennent pas à extraire la totalité des informations pertinentes. Cette limite peut se matérialiser, par exemple, par la définition de règles de décision non minimales (par exemple, elle est parfois atteinte par l'algorithme C4.5 (Quinlan, 1993)). De plus, l'extraction automatisée des connaissances, aussi efficace soit-elle, ne permet pas toujours de transmettre simplement ces connaissances à l'expert.

Pour résoudre ce problème, une solution viable consiste à présenter graphiquement les données de manière à permettre à la fois de guider, par l'interaction, une extraction de connaissances (automatique ou manuelle) et de permettre la transmission des connaissances à l'expert en l'aidant à construire sa carte mentale des données. Ainsi, on parlera de fouille visuelle de données ou de *visual data mining*. Le rôle de la visualisation dans la fouille visuelle de données peut se résumer en plusieurs points principaux (Fayyad et al., 2001). Elle permet d'afficher une grande quantité de données, fournissant de ce fait une vue d'ensemble des données. Elle

permet automatiquement, ou par interaction, de résumer et annoter des données La visualisation permet également de détecter des sous-ensembles de données où il est intéressant d'appliquer tel ou tel outil de fouille de données ou de fouille visuelle de données de manière plus précise. D'un autre côté, l'expert cherchant à analyser les données peut ne pas savoir ce qu'il cherche. Dans ce cas, le choix d'un algorithme de fouille de données est difficile à faire. La visualisation trouve alors son intérêt grâce à l'interaction, permettant à l'utilisateur d'établir une carte mentale des données et de rechercher des structures.

Que ce soit pour la confirmation d'hypothèses, pour la recherche de structures ou pour l'application d'algorithmes prenant en compte l'utilisateur, l'interaction de l'utilisateur avec la visualisation tient un rôle très important. L'interaction permet, entre autres, la navigation dans les données (vue sous différents angles), l'interrogation des données (réduction de l'affichage ou de caractéristiques d'affichage à une portion de l'espace) et l'association de données entre elles (marquages, simplification, réduction, ...) (Fayyad et al., 2001).

3.2 Pourquoi la visualisation de données a-t-elle un rôle si important ?

> « The history of visualization is that of the search for new artefacts to amplify the ability to know ; it's the history of writing and of maps, the history of knowledge. » (Dursteler, 2003)

Depuis des siècles, l'être humain utilise les représentations afin d'accroître sa capacité à intégrer des informations (Dursteler, 2003). Ces représentations prennent des formes très variées : écriture (chiffre, lettre, ...), cartes, graphiques, ... On peut alors se demander : « est-ce que l'homme a vraiment besoin de ces représentations ? » La réponse est très probablement « oui ».

Historiquement, le rôle des visualisations était de permettre la découverte et la compréhension de structures de hauts niveaux dans les données, impossibles à découvrir par une analyse directe des données (Eick, 2000). Prenons l'exemple d'une fonction f dont on connaît un certain nombre de valeurs. L'analyse de ces valeurs numériques fournit peu d'information sur la fonction f. Si, au lieu d'analyser ces valeurs, on trace le graphique correspondant aux valeurs de la fonction, il devient beaucoup plus facile de comprendre le comportement de la fonction. En effet, même en présence de bruit dans les valeurs, dû par exemple à des erreurs de calcul ou de précision de calcul, la représentation permet d'obtenir visuellement le comportement global de f. Par conséquent, la visualisation d'information permet de faciliter l'abstraction des informations dans les données.

Depuis quelques années, la visualisation d'information ne sert plus uniquement à la découverte d'information dans les données. Elle sert également de moyen d'accès aux détails des données et à l'affinement des caractéristiques statistiques extraites des données (Eick, 2000). Par conséquent, il devient de plus en plus courant de rencontrer des visualisations qui permettent de confirmer, ou d'expliquer des hypothèses définies *a priori*.

3.3 Définitions et structure générale d'un système de visualisation d'information

Pour décrire la structure générale d'un système de visualisation d'information, il est nécessaire d'introduire au préalable quelques définitions.

3.3.1 Définitions

Définition 8 *Une visualisation, c'est l'ensemble des moyens graphiques autres que ceux textuels ou ceux verbaux utilisés dans le but de communiquer des informations.*

Définition 9 *L'interaction regroupe l'ensemble des moyens mis à disposition de l'utilisateur et l'ensemble des actions de l'utilisateur permettant d'ajuster une visualisation de manière interactive.*

Définition 10 *Un attribut graphique est un aspect élémentaire d'une visualisation contrôlable par l'utilisateur définissant les caractéristiques de la représentation (position, taille, couleur, forme, orientation, vitesse, texture, transparence, ...).*

Définition 11 *La mise en correspondance ou mapping est l'opération qui consiste à définir la façon dont les attributs des données sont associés aux attributs graphiques .*

Définition 12 *La normalisation des données est l'opération qui consiste à ré-échelonner les valeurs prises par les différents attributs des données dans un intervalle donné.*

Il est possible de définir principalement trois types de normalisation (Fayyad et al., 2001) :
- la normalisation locale : chaque dimension est ramenée dans l'intervalle $[0, 1]$ ou $[-1, 1]$. Ce type de normalisation confère un poids égal à chacune des dimensions ;
- la normalisation globale : toutes les dimensions sont ré-échelonnées dans l'intervalle $[m, M]$ en désignant par m le minimum (respectivement M le maximum) des valeurs sur toutes les dimensions. Ce type de normalisation donne un poids plus important à la dimension qui a la plus grande valeur ;
- la normalisation pondérée : seules les dimensions qui ont un sens équivalent sont normalisées simultanément (ex : des kilogrammes de plomb et des kilogrammes de fer, ...).

De manière générale, la normalisation locale est la plus utilisée, car la plus neutre. Cependant, dans certains cas, les autres normalisations peuvent être plus efficaces pour la visualisation.

3.3.2 Structure générale d'un système de visualisation d'information

Le processus de visualisation d'information est un processus en cinq étapes (cf. figure 3.1) interagissant entre elles (Mroz et al., 1998) :
- les deux premières étapes sont l'acquisition et la préparation des informations. Après l'acquisition des données ayant pour origines des expérimentations, des observations, voire des expérimentations numériques ou des simulations, il est nécessaire de transformer ces données dans une forme compréhensible par la méthode de visualisation. Pendant la transformation, les données peuvent être filtrées, lissées, enrichies et/ou normalisées, afin d'améliorer la visualisation ;
- la troisième étape du processus consiste en la définition de la mise en correspondance entre les données et les éléments de la visualisation ;
- l'étape suivante consiste à traduire les données et la mise en correspondance en une représentation permettant l'analyse ;
- la dernière étape est l'analyse de l'utilisateur. Cette étape est cruciale, car c'est à ce moment que l'on permet à l'utilisateur d'ajuster les différents paramètres du processus de visualisation, afin de lui permettre d'explorer les données.

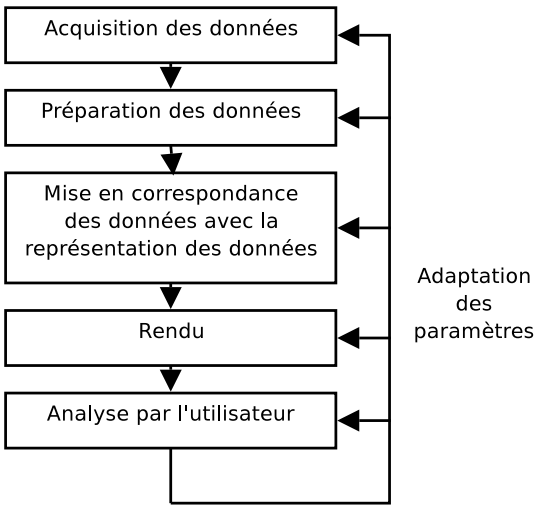


FIG. 3.1 – Processus de visualisation d'information.

3.4 Les difficultés rencontrées dans la visualisation d'information

La conception d'un système de visualisation d'information n'est pas aussi simple qu'elle pourrait le paraître au premier abord. En effet, pour être efficace, le système doit prendre en compte la dimensionalité des données et de la visualisation, ainsi que les limites physiques et psychologiques des sens de l'utilisateur.

3.4.1 Dimensionalité des données et de la visualisation

Que ce soit pour la représentation de fonctions multidimensionnelles, de nuages de points multidimensionnels ou de dissimilarité, la difficulté de la représentation et la représentativité qui en découle dépendent très fortement de la dimensionnalité intrinsèque des données N (Wijk and van Liere, 1993). En effet, pour $N = 1$, $N = 2$ ou $N = 3$, un simple rendu de graphe ou de volume avec des couleurs en 2D ou en 3D suffit, bien qu'il ne soit pas toujours le mieux adapté. Pour $N = 4$, on peut parfois se ramener aux cas $N = 3$, en considérant qu'une des dimensions est une dimension temporelle, mais, pour $N > 4$, il n'est en général plus possible de représenter l'intégralité des N dimensions sans les déformer. Trois principales techniques sont utilisables, conjointement ou non :

- on fixe la valeur de certaines dimensions, et on représente les autres dimensions (ex : le *scatterplot*, HyperSlice, ...),
- on déforme la représentation de manière à tenir compte de l'intégralité des dimensions (ex : radviz, les projections, ...),
- on utilise des artifices de représentation pour augmenter le nombre d'attributs visualisables (ex : couleur, forme, direction, ...). L'utilisation de cette dernière technique est réalisée très couramment, mais cela nécessite de prendre des précautions pour qu'elle soit efficace (cf. section 3.4.2).

3.4.2 La perception des sens dans la visualisation d'information

La construction d'une visualisation nécessite de prendre en compte, au maximum, quatre aspects de la perception des sens : la capacité de jugement, la mémorisation, le niveau d'abstraction des représentations et le contexte visuel. La figure 3.2 synthétise ces quatre aspects.

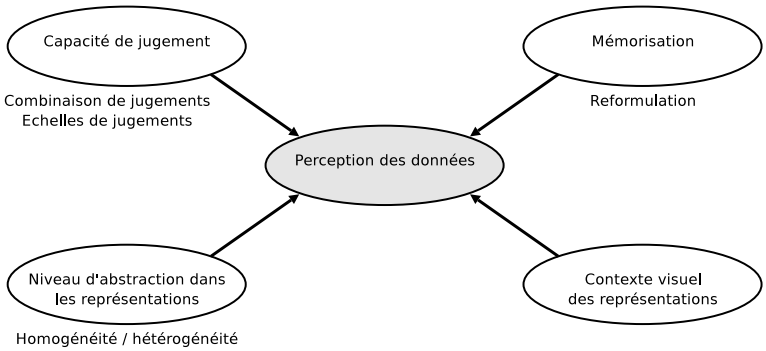


FIG. 3.2 – Les quatres aspects principaux qui influent sur la perception des données.

3.4.2.1 La capacité de jugement

Plusieurs études faites par G. Miller (1956) (Fayyad et al., 2001) ont eu pour objectif l'étude des aptitudes de l'homme à prendre en compte des stimuli. Les études portant sur le jugement absolu de stimuli sonores (pitch, force), gustatifs (salinité) et visuels (positionnement sur une ligne, taille de carrés, longueur, orientation et courbure de lignes) ont permis de montrer que l'homme était capable de distinguer approximativement 10 niveaux différents à chaque fois. G. Miller a également effectué une étude en utilisant plusieurs stimuli en simultané, il a alors constaté une augmentation des capacités de jugement, mais cette capacité n'est pas le produit des capacités des différents stimuli. Ce résultat est confirmé dans (Fayyad et al., 2001) où est évoquée une expérience impliquant la taille, la couleur et la luminosité de carrés de capacités respectives de jugement 5, 8 et 5. La capacité de jugement absolue des stimuli, lorsqu'ils sont combinés, est alors de 17 et non pas 5*8*5=200. (Baker and Wickens, 1995) souligne le fait que les ressources utilisées pour l'analyse auditive et visuelle sont différentes et que les ressources impliquées dans l'analyse des formes, des couleurs et des orientations sont également différentes.

Cleveland (Baker and Wickens, 1995) s'est également intéressé à l'étude de la perception des stimuli. Il est arrivé à la conclusion que la capacité de l'homme à effectuer des jugements relatifs est significativement supérieure à la capacité d'effectuer des jugements absolus de stimuli. De plus, Cleveland indique, par ordre décroissant de capacité, que les tâches consistant à détecter des changements relatifs sont :

- positionnement sur une échelle commune,
- positionnement sur des échelles communes non alignées,
- repérage des longueurs,
- repérage des angles et des inclinaisons (dépend beaucoup de l'orientation et du type de l'objet affiché),
- repérage des aires,
- repérage des volumes,
- repérage des couleurs : teinte, saturation, densité.

D'un autre côté, la loi de Stevens (Fayyad et al., 2001) souligne le fait que l'échelle (visuelle, auditive, ...) est très importante. En effet, pour un stimulus visuel, la perception dans un jugement absolu de la taille pour un attribut graphique linéaire est approximativement la taille réelle élevée à une puissance comprise entre 0.9 et 1.1. Pour une aire, la puissance considérée se situe dans l'intervalle $[0.6, 0.9]$ et celle pour un volume est dans l'intervalle $[0.5, 0.8]$. Cette loi implique donc que la notion d'échelle absolue est mieux traduite par un attribut graphique linéaire que par une aire ou un volume.

3.4.2.2 La mémorisation et la reformulation

Comme il a été évoqué plus haut, le nombre moyen de stimuli perceptibles par l'homme est approximativement de 10. Pour augmenter le nombre d'informations qu'un individu peut prendre en compte, il est nécessaire de faire intervenir la mémoire immédiate. En moyenne, la mémoire immédiate permet à l'homme de conserver 7 informations (plus ou moins 2). Chaque information peut être de plusieurs types : une valeur simple, un stimulus, une combinaison de stimuli ou quelque chose de plus complexe. Pour permettre à chaque case mémoire de retenir une information complexe, la personne a besoin de recoder les informations qu'elle cherche à mémoriser. Ce recodage peut prendre plusieurs formes (une image, un symbole, une phrase, une association, ...). Il est donc parfaitement possible d'affirmer que dans une visualisation, quelle qu'elle soit, la prise en compte de l'activité de reformulation par l'utilisateur est importante. Cependant, dans (Fayyad et al., 2001), il nous est rappelé que ce recodage des informations est propre à chaque individu et que, par conséquent, des variations tant au niveau de la personne qu'au niveau du contexte dans lequel il est fait sont très probables. Une visualisation doit donc s'efforcer de favoriser cette reformulation, mais elle ne doit pas l'imposer.

3.4.2.3 Le niveau d'abstraction des représentations

D'un autre côté, (Baker and Wickens, 1995) rappellent que la décidabilité sur une représentation visuelle dépend fortement du niveau d'abstraction du symbole utilisé dans la visualisation. La loi de Gestalt établit que les formes fermées, régulières et symétriques impliquent une forte unicité de la perception de la forme (ex : $<>$ est perçu comme un unique objet tandis que $>\hat{}$ ne l'est pas). Cependant, l'encodage de plusieurs variables dans un unique *glyph*[1] facilite l'intégration des attributs, mais cette intégration rend difficile l'action de focalisation sur une des caractéristiques, tout en ignorant les autres. De plus, la comparaison de deux ensembles de données n'est possible que s'ils sont codés sur un même ensemble de caractéristiques, car l'homogénéité améliore la découverte de lien ou de différence entre les données.

3.4.2.4 Le contexte visuel

L'arrangement spatial dans la visualisation joue également un rôle important (Baker and Wickens, 1995). En effet, si les données ont une organisation géographique intrinsèque, la compréhension est améliorée en la conservant. Cependant ce choix laisse moins d'attributs graphiques utilisables pour les autres attributs . D'autres facteurs permettent d'améliorer la perception d'une visualisation : les ombres fournissent les informations sur la hauteur et la distance au sol, la perspective informe sur la profondeur, les grilles et les marqueurs permettent de clarifier la position des objets de la scène tandis que la stéréoscopie améliore les temps de découverte de connaissances à partir des données. Il est aussi important de ne pas encombrer une visualisation avec trop de détails car l'homme n'est pas capable de tous les appréhender efficacement et cela peut même nuire à la compréhension. Il est donc nécessaire de pouvoir identifier les objets importants par des marqueurs. Il est aussi important de savoir que les textures, les gradients et la vue stéréoscopique ont moins d'importance que la lumière et les ombres dans une représentation.

Finalement, le rôle d'une visualisation d'information consiste à définir une carte mentale des données. Cette carte mentale sert de moyen d'encodage de la position relative des objets les uns par rapport aux autres et de moyen de codage des attributs des objets dans l'environnement.

[1] Le terme anglais *glyph* représente une forme en général.

3.5 Les techniques de visualisation

Depuis ses débuts, le domaine de la visualisation d'information n'a cessé d'inspirer de nombreux travaux. Ces travaux ont donné naissance à de nombreuses techniques de représentation de l'information. Certaines d'entre elles ont particulièrement marqué le domaine et ont inspiré de nombreuses variantes plus ou moins adaptées à des domaines particuliers. Dans cette section, nous n'allons pas tenter de dresser un panorama exhaustif des différentes techniques existantes ni des différents logiciels les implémentant. Nous allons nous attacher à la description des principales techniques de visualisation de l'information indépendamment des logiciels ou « pack logiciel » dans lesquelles elles sont implémentées. Dans cette étude, nous avons volontairement ignoré les techniques de visualisation trop spécifiques, telles que la visualisation médicale, la visualisation de volume ou de force, ...afin de nous concentrer sur la visualisation de graphes, de données multidimensionnelles et de dissimilarité/similarité. Le lecteur intéressé par une bonne introduction aux techniques et problèmes principaux de la visualisation d'information pourra se référer à (Card et al., 1999).

3.5.1 Le graphe de courbes

Historiquement, le graphe de courbes (*line graphe*) est la première visualisation qui a été utilisée. Son principe est simple, il consiste en la représentation habituelle de fonction continue $y = f(x)$. Bien que ne supportant que deux dimensions, cette représentation peut être étendue en superposant plusieurs graphes (Fayyad et al., 2001). Ainsi, en superposant n graphes en utilisant le même paramètre x, on représente un total de $n + 1$ dimensions de l'espace. Ce type de graphe est appelé graphe de courbe multiple (*multiple line graph*) (cf. figure 3.3).

Cependant, comme la dimension x joue un rôle particulier (les valeurs ne se répètent pas et x a le sens de « paramètre » des données), cette particularité peut parfois poser problème lors de l'interprétation. De plus, il peut être nécessaire d'utiliser des échelles différentes en ordonnée et des *offset* de décalages rendant la représentation confuse le plus souvent à partir de trois courbes.

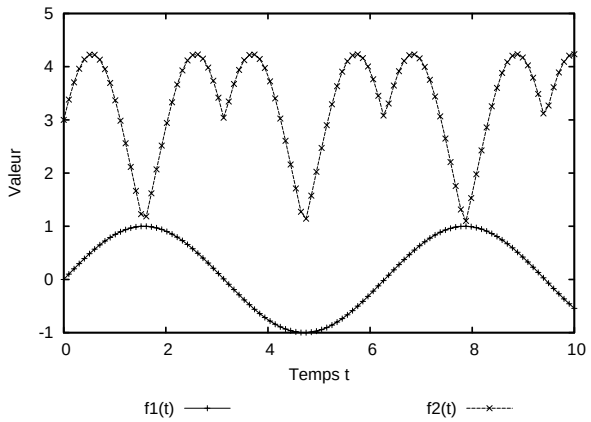


FIG. 3.3 – Un exemple de graphe de courbes.

3.5.2 Le *scatterplot*, la matrice de *scatterplots* et ses dérivées

3.5.2.1 Le *scatterplot*

Le nuage de points ou *scatterplot* est un graphe en deux dimensions affichant un ensemble de couples (x, y) sous la forme d'un nuage de points.

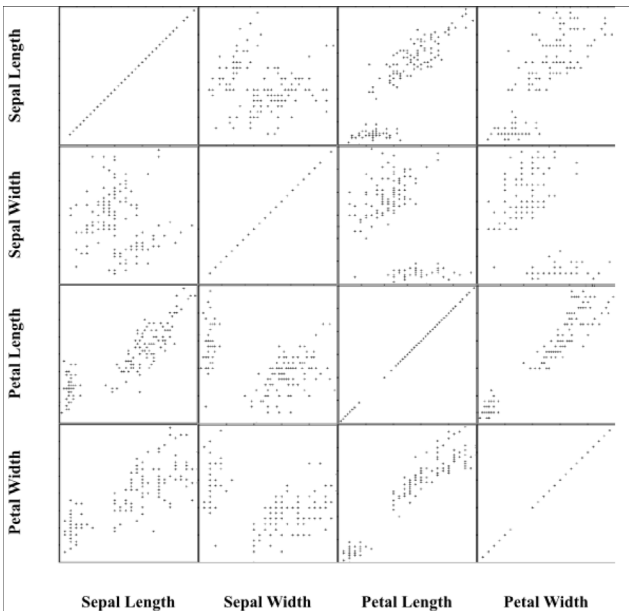


FIG. 3.4 – Exemple de matrice de *scatterplots* sur la base Iris (Fisher, 1936).

3.5.2.2 La matrice de *scatterplots*

La matrice de *scatterplots* (Cleveland, 1993) (Wong and Bergeron, 1997) (cf. figure 3.4) est, comme son nom l'indique, une matrice de plusieurs *scatterplots* dont les axes changent. Si on considère $(x_1, x_2, \ldots, x_n)$ les dimensions dans lesquelles sont décrites les données, alors la matrice est obtenue en énumérant les paires de dimensions et en traçant le *scatterplot* associé pour chaque paire (cf. figure 3.4). Comme on peut le remarquer, la matrice est symétrique et la diagonale[2] haut/gauche vers bas/droite représente l'étalement des valeurs de la coordonnée.

L'un des principaux avantages de la matrice de *scatterplots* est de permettre la représentation de données discrètes sans donner un rôle particulier à une dimension de l'espace (Wijk and van Liere, 1993), contrairement aux graphes de courbes. Les matrices de *scatterplots* sont très populaires pour l'analyse de données, car elles permettent la recherche des corrélations entre deux dimensions, de classe, de tendance et de données incorrectes.

La matrice de *scatterplots* permet également l'utilisation de brosses (*brushing*[3]), la coloration des classes, le zoom, le *jittering*[4]. Elle a aussi été utilisée avec des *glyphs*, des icônes et des couleurs (Fayyad et al., 2001).

Bien que le nombre de dimensions utilisables avec une matrice de *scatterplots* soit limité par la taille de l'écran, il est possible d'en augmenter la capacité de représentation en utilisant une représentation 3D (Scullin et al., 1995), des animations, des couleurs, des formes et l'interaction, afin de montrer pour chaque *scatterplot* jusqu'à 3, 4, 5 ou plus de dimensions (Fayyad et al., 2001).

[2]Dans certaines implémentations, les *scatterplots* ne sont pas disposés de la même façon. La diagonale est alors du coin haut/droite vers le coin bas/gauche. Par exemple, dans la technique de la matrice de *scatterplots* pseudo-euclidienne que nous proposons au chapitre 6, nous adoptons cette disposition alternative.

[3]L'utilisation de brosses ou *brushing* est le terme consacré aux techniques permettant la sélection de certains points et le filtrage interactif de ces dernières. (Martin and Matthew O. W., 1995) présente plusieurs modes de *brushing*, implémentés dans le logiciel XmdvTool, pour la matrice de *scatterplots*, les *parallel coordinates* et le *dimensional stacking*.

[4]Le *jittering* est la technique consistant à ajouter un bruit aléatoire aux projections de manière à rendre visible des points ayant les mêmes coordonnées de projection.

3.5.2.3 La technique HyperSlice

La représentation en HyperSlice (Wijk and van Liere, 1993) est une méthode de visualisation dérivée des matrices de *scatterplots* afin de permettre la visualisation de fonction scalaire de plusieurs variables. Par fonction scalaire, on entend les fonctions du type $f(x) = y$ avec x une donnée multidimensionnelle et y un réel. Pour adapter la représentation en *scatterplot*, les auteurs définissent un point courant c de l'espace et un ensemble d'intervalles sur chacune des dimensions autour de c. Chaque *scatterplot* de la matrice est obtenu en sélectionnant deux dimensions x_i et x_j et en dessinant en niveaux de gris la valeur de la fonction en faisant varier x_i et x_j dans les intervalles spécifiés et en laissant les autres dimensions aux valeurs des dimensions du point c.

3.5.3 Les techniques de projection

3.5.3.1 La technique *Radviz*

La méthode de visualisation *Radviz* (pour visualisation radiale) (Hoffman et al., 1997) (Brunsdon et al., 1998) est une technique de projection utilisant la notion d'ancre de dimensions, comme expliqué dans (Hoffman et al., 1999).

La technique *RadViz* consiste à définir sur un cercle unitaire m points d'ancrage correspondant chacun à une des m dimensions des données, de manière à ce qu'ils soient régulièrement espacés. La projection d'une donnée sur le plan correspond au point d'équilibre annulant la somme des forces exercées sur ce point vers les points d'ancrage. La force s'exerçant entre le point d'équilibre et un point d'ancrage a pour direction le vecteur ayant pour origine le point d'équilibre et pour extrémité le point d'ancrage (cf. figure 3.5 (a)). La force s'exerçant est proportionnelle à la valeur prise par la donnée sur cette dimension.

Le point d'équilibre u_i relatif à la donnée $x_i = (x_{i,1}, \ldots, x_{i,m})$ pour les points d'ancrage $S_{j/j=1..m}$ est la solution de l'équation

$$\sum_{j=1..m} (S_j - u_i)x_{i,j} \quad = \quad 0$$

c'est-à-dire

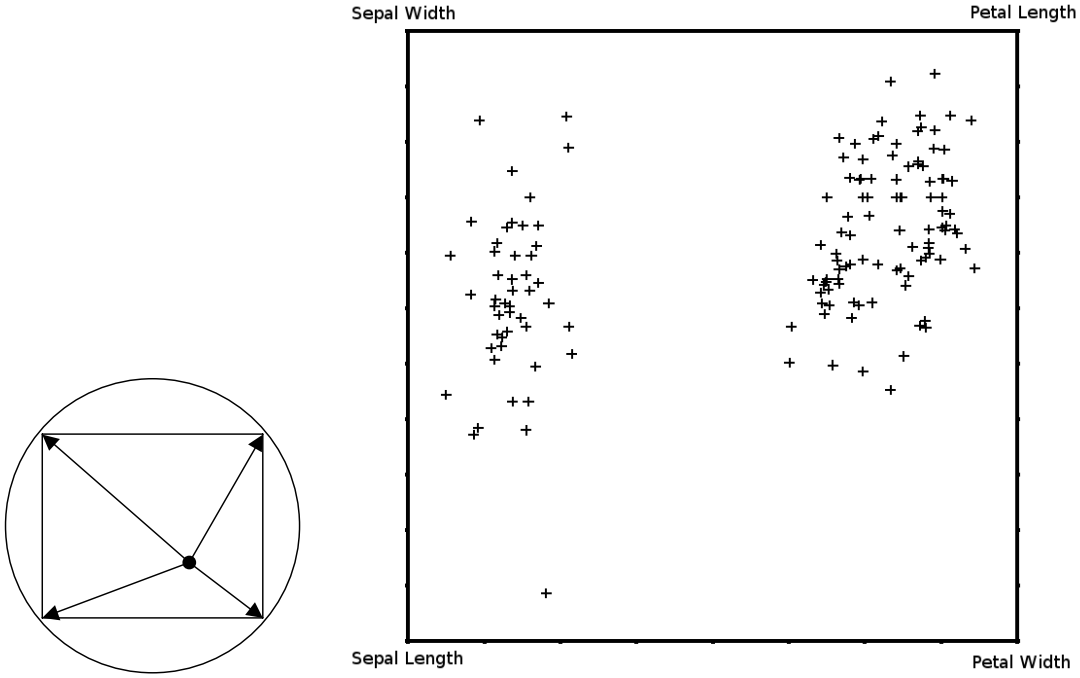
$$u_i \quad = \quad \sum_{j=1..m} \frac{x_{i,j}}{\sum\limits_{k=1..m} x_{i,k}} S_j$$

Ce point d'équilibre est donc simplement le barycentre des points d'ancrage avec pour masse

$$\frac{x_{i,j}}{\sum\limits_{k=1..m} x_{i,k}}$$

De plus, cette représentation assure qu,e si $x_{i,j} \geq 0$, alors les projections des points sont contenues dans le polygone formé par les points d'ancrage (Brunsdon et al., 1998). Cette condition est facilement réalisable par une étape préliminaire de normalisation des données. La figure 3.5 (b) montre le résultat de la projection d'un jeu de données.

Cette méthode possède pour principal inconvénient sa dépendance à l'ordre des points d'ancrage sur le cercle unitaire. En effet, il est tout à fait possible qu'un ordre particulier fasse que des forces s'annulent mutuellement si leurs points d'ancrage sont diamétralement opposés sur le cercle. En éliminant les configurations d'ancres similaires, à une rotation et une symétrie près, il existe exactement $\frac{(m-1)!}{2}$ projections différentes envisageables (Brunsdon et al., 1998).



(a) (b)

FIG. 3.5 – Visualisation par la technique *RadViz* : (a) les forces s'exerçant sur un point d'équilibre avec 4 ancres, (b) Exemple de projection de la base Iris (Fisher, 1936).

3.5.3.2 Les cartes auto-organisatrices de Kohonen

Les *Self Organizing Maps* (SOM) (Vesanto, 1997) (Vesanto, 1999), autrement nommées cartes auto-organisatrices de Kohonen, du nom du professeur qui les a développées au début des années 80, sont fondées sur un apprentissage neuronal non supervisé et concurrentiel des données. De par leurs organisations spatiales, les SOM sont des outils classiques de projection et de quantification[5] simultanées.

Les SOM. sont formées, pour des données multidimensionnelles en n dimensions, de neurones uniformément répartis, principalement, sur une grille 1D ou 2D. Chacun des neurones est muni d'un vecteur de n poids. Les neurones sont reliés aux neurones constituant leurs voisinages (plusieurs types de voisinages peuvent être considérés, comme par exemple des voisinages rectangulaires, hexagonaux, ...). Le nombre de neurones et la taille du voisinage déterminent la finesse de l'analyse. Après une initialisation des vecteurs poids aléatoirement, par échantillonnage des données, par des vecteurs propres ou par toutes autres méthodes, les données sont apprises par le réseau. A chaque itération de l'apprentissage, on choisit aléatoirement un vecteur x parmi les données et on calcule sa similarité avec les vecteurs poids des neurones. On détermine alors le neurone ayant les poids les plus similaires à la donnée (*Best Matching Unit* ou BMU). On met alors les vecteurs poids de la BMU et du voisinage à jour en suivant l'équation suivante (Vesanto, 1999) :

$$m(t+1) = m(t) + h(t)(x - m(t))$$

en désignant par $m(t)$ le vecteur poids à mettre à jour, x la donnée, $h(t)$ la fonction noyau du voisinage pour le vecteur. La fonction $h(t)$ est décroissante en fonction du temps t et de la distance du neurone à la BMU. Cet apprentissage a pour effet d'attirer les neurones de la BMU et du voisinage vers la donnée. La figure 3.6 en schématise le principe.

[5]Définition des données en fonction d'un *codebook* discret.

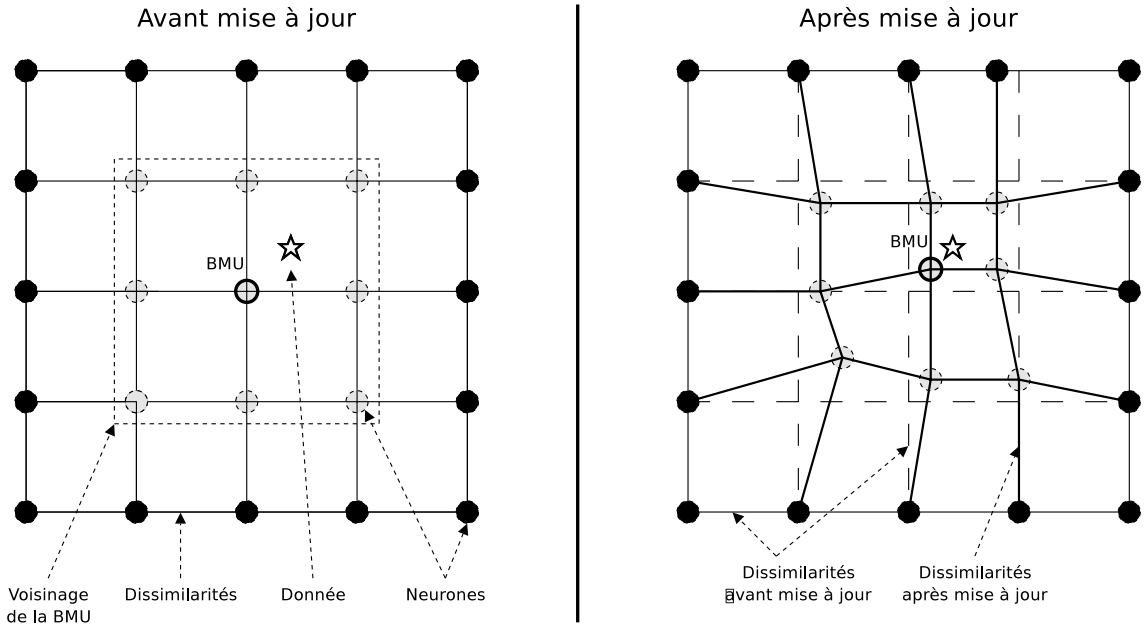


FIG. 3.6 – Schématisation de l'attraction des neurones vers une donnée lors d'une itération de l'algorithme des cartes auto-organisatrices (figure adaptée à partir de (Vesanto, 1997)).

Comme nous l'avons déjà dit plus haut, les SOM réalisent une quantification et une projection des données, c'est-à-dire qu'elles cherchent de bons vecteurs pour servir de représentants des données et en plus elles ordonnent ces vecteurs sur la grille. Les SOM ont également la propriété d'éliminer le bruit dans les données (Vesanto, 1999). Cependant, contrairement aux méthodes de projection standard, les SOM ne conservent pas les distances entre les données directement, elles se contentent de conserver la structure locale des données et, par conséquent, l'interprétation des SOM ne doit se faire que localement.

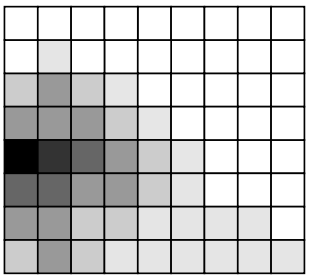


FIG. 3.7 – Exemple de visualisation d'une composante des vecteurs associés aux neurones d'une carte auto-organisatrice en niveaux de gris.

La visualisation des SOM revêt plusieurs aspects :

- la visualisation des composantes des vecteurs de poids des neurones. On visualise en niveaux de gris la valeur de chaque composante sur une carte comme dans la figure 3.7 (Vesanto et al., 1998). Il est de plus possible de rechercher des corrélations entre les diverses composantes des vecteurs de poids en adoptant une visualisation simultanée de plusieurs composantes.
- la visualisation de l'organisation des groupes (*clusters*) dans les SOM. On visualise une matrice de similarité des vecteurs poids avec leur voisinage en niveau de gris.
- la visualisation de la projection des vecteurs de poids. On effectue une projection des

vecteurs poids des neurones par une projection en 2D ou 3D préservant les distances (cf. section 3.5.3.3 sur le *Multidimensional Scaling*), comme la projection de Sammon, et on relie les neurones voisins. Ce type de visualisation permet d'appréhender la structure de la SOM.
- une autre visualisation consiste à comptabiliser le nombre de fois qu'un neurone est choisi comme BMU par les données sous forme de rectangle de taille correspondante ou d'histogramme.
- lorsque les données ont un ordre séquentiel naturel, on peut visualiser la séquence des BMU en les reliant.

Le lecteur désirant voir d'avantage de représentations graphiques des SOM pourra se référer à (Vesanto, 1997) et (Vesanto, 1999).

3.5.3.3 *Multidimensional scaling*

Le multidimensional scaling (MDS) (Borg and Groenen, 1997) (Cox and Cox, 2001), rarement nommé « échelonnement multidimensionel » en langue française, est un ensemble de techniques qui consistent à projeter un nuage de points connus par leurs proximités respectives dans un espace donné. Le MDS peut à la fois être considéré comme un pré-traitement et comme une méthode de visualisation. On parlera de pré-traitement lorsqu'il s'agit de projeter le nuage dans un espace de dimension quelconque, et de visualisation si la projection s'effectue dans un espace de faible dimension (ex : 1D, 2D ou 3D) avec l'objectif de voir le nuage. Pour effectuer cette projection, le MDS cherche à minimiser une fonction de *stress* dépendant des proximités *a priori* entre les données ($p_{i,j}$) et des proximités effectives de leurs représentants dans l'espace projeté ($d_{i,j}$). De nombreuses fonctions de *stress* ont été étudiées au fil des années (Borg and Groenen, 1997) (Cox and Cox, 2001). Les techniques fondamentales sont décrites dans le tableau 3.1.

Nom	Objectif
classical MDS	$p_{i,j} = d_{i,j}$
interval MDS	$a + bp_{i,j} = d_{i,j}$
ordinal MDS	$p_{i,j} < p_{k,l} \Longrightarrow d_{i,j} \leq d_{k,l}$
MDS généralisé	$f(p_{i,j}) = d_{i,j}$ en désignant par f une fonction réelle monotone

$d_{i,j}$ peut être définie de plusieurs façons. De manière classique, $d_{i,j}$ est une distance de Minkowski ($d(x,y) = \left(\sum_t |x_t - y_t|^K\right)^{1/K}$) mais elle peut également être une distance sur un cercle ou toute autre mesure.

TAB. 3.1 – Techniques fondamentales de *multidimensional scaling*.

3.5.4 La visualisation de graphes

La visualisation de graphes représente tout un pan de la visualisation d'information. En effet, de nombreuses techniques y sont dédiées. Les différentes visualisations se distinguent les unes des autres suivant quatre critères : le type de graphe représenté (arbre, graphe complet, orienté, non orienté, ...), la taille du graphe, le placement des noeuds du graphe et la représentation des noeuds et des arcs entre les noeuds.

La technique *Nicheworks* (Wills, 1999) en est un bon exemple. Cette technique est principalement destinée à la représentation d'arbres. Pour cela, elle adopte une stratégie de division[6] récursive de l'espace en fonction de la taille des sous-arbres (cf. figure 3.8).

[6]Il s'agit d'une division d'angles.

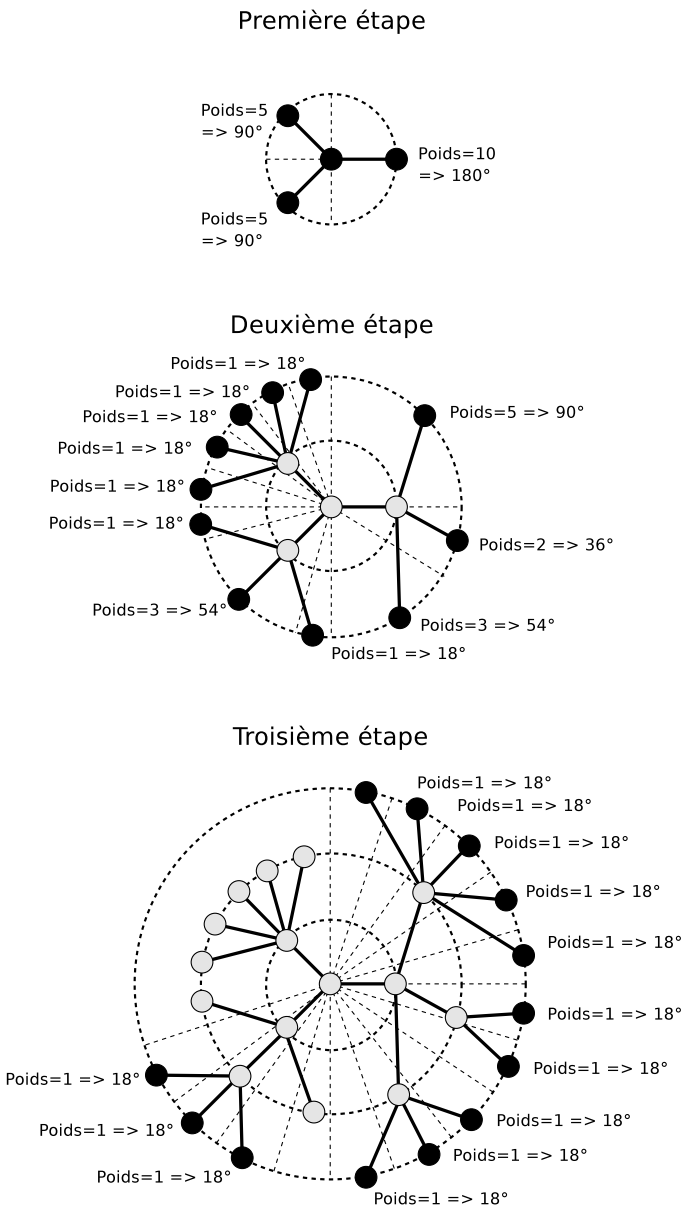


FIG. 3.8 – Principe de positionnement/partitionnement de la technique *Nicheworks* (figure adaptée de (Wills, 1999)).

Étant donnée la taille de ce sous-domaine de la visualisation d'information, nous n'allons pas effectuer un inventaire des principales techniques de la visualisation de graphes. Cependant, nous encourageons le lecteur intéressé à se référer aux articles (Di Battista and Eades, 1994) et (Herman et al., 2000) établissant une bonne revue des techniques existantes dans le domaine. D'autres techniques connues, telles que les *Tree-maps* (Johnson and Shneiderman, 1991) pour des hiérarchies ou la technique du *Cone-tree* (Robertson et al., 1991), y sont référencées.

3.5.5 Les matrices de similarité en niveaux de gris

Les matrices de similarité en niveaux de gris ou *shaded similarity matrix*, également connues sous les noms de *shaded proximity matrix* et *trellis diagram*, sont un moyen de visualiser des similarités entre des éléments. Les similarités sont matérialisées par un niveau de

couleur variant du plus clair au plus foncé, respectivement pour les faibles similarités et les fortes similarités (Wang et al., 2002b) (Wang et al., 2002a). La figure 3.9 donne un exemple d'une telle matrice.

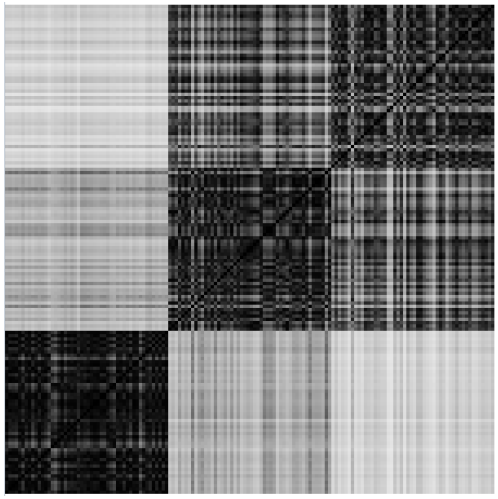


FIG. 3.9 – Exemple de matrice de similarité en niveaux de gris de la base Iris (Fisher, 1936). Les niveaux de gris sombres marquent une forte similarité entre les données.

Cette représentation des données est très efficace pour détecter des groupes de données, ces derniers apparaissant alors sous forme de carrés foncés et symétriques sur la diagonale. Comme il existe de nombreuses façons d'arranger les éléments pour la représentation ($n!$ pour n objets), on utilise généralement une méthode d'organisation des objets dans le but de faire ressortir les caractéristiques des données. Plusieurs types de méthodes existent. Beaucoup cherchent à trouver une organisation des données minimisant un critère, comme la variation locale des niveaux de similarité. D'autres appliquent une organisation du type *concept tree based ordering* consistant à construire un arbre de décision, et à organiser les données afin que la représentation respecte l'arbre de décision (Wang et al., 2002b). Ce type d'organisation peut être préférable, car la structure sous-jacente de l'arbre de décision rend l'interprétation de la matrice plus facile.

3.5.6 *Parallel Coordinates* et ses variantes

3.5.6.1 *Parallel Coordinates*

La technique des *parallel coordinates* (fig. 3.10), initialement introduite par A. Inselberg au début des années 1980, consiste à représenter chaque point du jeu de données par une polyligne[7] (Inselberg and Dimsdale, 1990) (Inselberg, 2002). Pour cela, on considère n axes parallèles (n est la dimension des données à représenter) régulièrement espacés. Chaque axe correspond à une dimension des données. Un point des données est alors représenté par la polyligne reliant les différents axes. L'intersection de la polyligne avec un axe correspond à la valeur de la dimension associée à l'axe pour cette donnée. Si les axes parallèles sont aux coordonnées d'abscisse $1..n$, alors le point $x = (x_1, x_2, \ldots, x_n)$ est représenté par la polyligne passant par les points de coordonnées $(1, x_1), (2, x_2), \ldots, (n, x_n)$. Cette méthode transforme donc chaque point de $\mathbb{R}^n$ en $(n-1)$ segments joignant n lignes parallèles.

[7] Une polyligne est le résultat de la mise bout à bout de plusieurs segments.

Cette transformation des points possède des propriétés remarquables très intéressantes (Inselberg and Dimsdale, 1990) (Chou et al., 1999) (Inselberg, 2002) :

- une droite en dimension n peut s'écrire sous la forme de $n-1$ équations $x_{i+1} = m_i x_i + b_i \; \forall \; i = 1..n-1$;
- les points situés sur une droite de $\mathbb{R}^n$ sont transformés en un ensemble de polylignes dont chaque groupe de segments s'intersecte au point de coordonnées $\left(\frac{i}{1-m_i}, \frac{b_i}{1-m_i}\right) \; \forall \; i = 1..n-1$ si $m_i \neq 1$;
- quand $m_i = 1$, la droite d'angle directeur $\frac{b_i}{i}$ joignant un point situé à l'infini caractérise la relation linéaire (ce n'est plus un point d'intersection mais une direction) ;
- la rotation ou la translation d'une ligne en dimension n par rapport à un point se traduit respectivement par une translation et une rotation des points d'intersections des groupes de segments ;
- des lignes parallèles en dimension n sont alors transformées en polylignes dont les points d'intersections sont alignés verticalement ;
- par conséquent, un hyperplan en dimension n est représenté par plusieurs alignements verticaux de points d'intersection.

Cette méthode supporte classiquement un certain nombre d'interactions et d'outils (Hauser et al., 2002) :

- le réordonnancement des axes,
- le renversement des axes *i.e.* le retournement vertical de l'axe,
- l'utilisation d'une normalisation locale ou globale pour chacun des axes,
- le changement de la longueur des axes,
- l'ajout et la suppression d'axes,
- les détails sur une donnée à la demande,
- les brosses de sélection (*brushing*) :
  - les brosses angulaires permettant la sélection et le filtrage de points grâce à un critère de relation d'ordre sur deux de ses coordonnées.
  - les brosses composites sont obtenues par combinaison via des opérateurs logiques d'autres brosses. Elles peuvent être à valeur booléenne ou continue, grâce à l'utilisation d'opérateurs de logique floue.
- les histogrammes des valeurs prises sur les axes permettent de mieux appréhender la distribution des données par rapport à chaque axe. Ces histogrammes sont affichés en semi-transparence, afin de ne pas perturber trop la relation visuelle décrite par les polylignes.

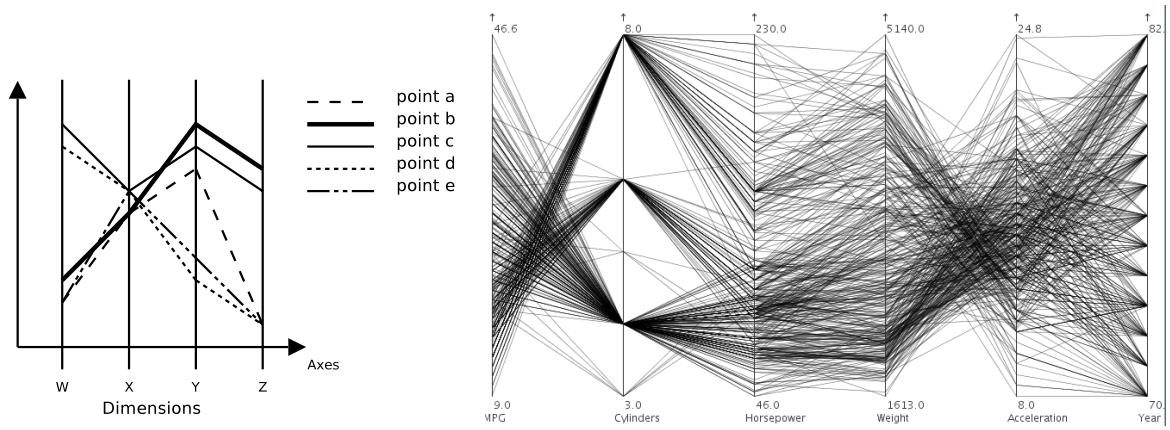


FIG. 3.10 – Deux exemples d'applications de la technique de *Parallel Coordinates*. Le deuxième exemple a été obtenu à l'aide du logiciel Parvis (Ledermann, 2002) sur la base de données *Car* (Newman et al., 1998).

Un certain nombre d'extensions de la technique des *Parallel Coordinates* ont été développées.

3.5.6.2 *3D Parallel Coordinates*

3D Parallel Coordinates (Mroz et al., 1998) : une généralisation des *Parallel Coordinates* dans laquelle les axes parallèles sont remplacés par des plans parallèles. Une donnée est représentée par une polyligne joignant des plans parallèles. Chacun des plans correspond à deux coordonnées. Par exemple, si deux plans parallèles représentent les dimensions (x_1, x_2) et (x_3, x_4), le point $(0, 0, 1, 1)$ sera représenté par le segment joignant le point de coordonnées $(0, 0)$ du premier plan et le point de coordonnées $(1, 1)$ du deuxième plan.

3.5.6.3 *Hierarchical Parallel Coordinates*

Hierarchical Parallel Coordinates (Fua et al., 1999) : dans cette extension, les données sont d'abord classées au sein d'une hiérarchie. Les représentants des classes à un certain niveau de la hiérarchie sont visualisés grâce à la technique des *Parallel Coordinates*. Afin de représenter la taille et la variance des classes, les polylignes sont élargies à une largeur dépendante de la taille de la classe. Ces polylignes deviennent transparentes au fur et à mesure que l'on s'éloigne du représentant de la classe.

3.5.6.4 *Circular Parallel Coordinates*

Circular Parallel Coordinates (Hoffmann and Grinstein, 2005) : cette extension correspond à un positionnement en étoile des axes parallèles de la technique de *Parallel Coordinates* (cf. figure 3.11). L'intérêt principal de cette technique est que chaque axe joue le même rôle.

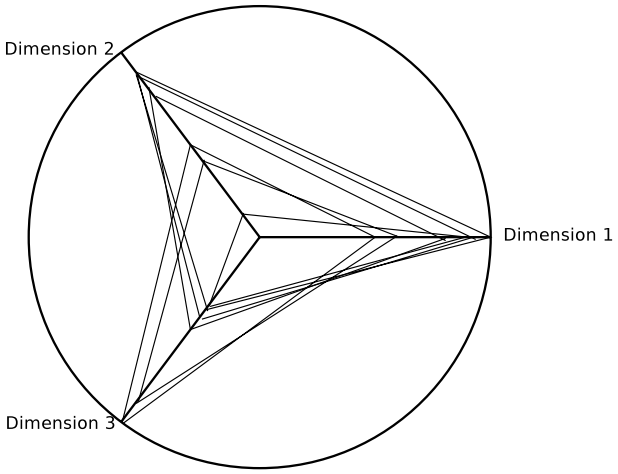


FIG. 3.11 – Exemple d'utilisation de la technique de *Circular Parallel Coordinates* sur un jeu de données fictifs.

3.5.7 L'empilement de dimensions

L'empilement de dimensions ou *dimensional stacking* est une technique générale permettant de représenter sans perte d'information des données multidimensionnelles tout en minimisant les occlusions. Pour cela, une stratégie récursive est adoptée dans la construction de la représentation. Plusieurs variations sur le principe du *dimensional stacking* ont été réalisées dans la littérature. Nous allons décrire rapidement les trois plus connues : *N-Land*, *co-plot* et *Worlds within Worlds*.

3.5.7.1 *N-land*

La technique *N-land* (Ward et al., 1994) considère des données à valeurs discrètes codées sur un nombre impair de dimensions[8]. On applique récursivement la méthode suivante : on choisit chacune des dimensions l'une après l'autre, en considérant que la dimension choisie dépend uniquement des dimensions précédentes. En prenant les dimensions deux par deux, on construit alors un graphe contenant pour chaque coupe de coordonnées, un autre graphe. Ce graphe correspond aux deux dimensions suivantes et ce processus est itéré jusqu'à épuisement de toutes les dimensions. La dernière dimension est représentée par un niveau de gris de manière à ce que chaque point corresponde aux coordonnées imbriquées de tous les graphes. La figure 3.12 schématise ce processus. Cette technique permet de détecter et d'analyser des relations linéaires impliquant jusqu'à 4 dimensions.

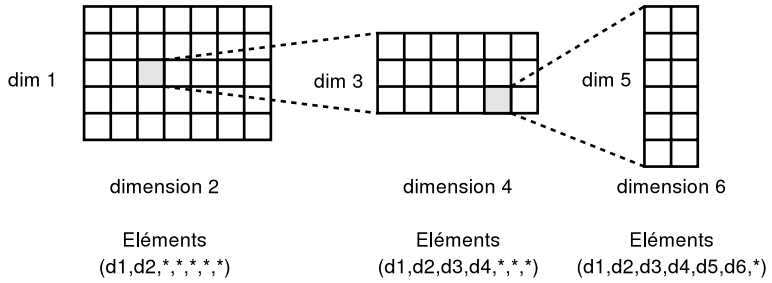


FIG. 3.12 – Structure d'imbrication pour la construction d'une représentation *N-land* pour 7 dimensions.

3.5.7.2 *Co-plot*

La technique de *co-plot* est similaire au *dimensional stacking*. Dans (Carr, 1995), l'auteur utilise un *co-plot* pour rechercher un minimum local d'une fonction de 4 variables. La technique de *co-plot* consiste à définir une grille sur 2 variables, et dans chacune des zones de ces grilles à dessiner le résultat de la fonction en faisant varier les 2 autres variables. Dans l'article, les valeurs de la fonction sont représentées par des angles formés par un petit segment par rapport au point d'ancrage. C'est dans cette variation de la présentation du résultat que se trouve la différence avec *N-Land*.

3.5.7.3 *Worlds within Worlds*

Le principe de la technique *Worlds within Worlds* (Feiner and Beshers, 1990) (Wong and Bergeron, 1997) est simple. Pour l'expliquer, prenons un exemple. On considère une fonction f de six variables $f(x_1, x_2, x_3, x_4, x_5, x_6)$. Représenter les valeurs prises par une telle fonction est difficile. Un moyen simple consiste à fixer trois variables parmi les six et à représenter le nuage de points correspondants. C'est ce que fait la technique *Worlds within Worlds*. Elle considère un graphe principal en trois dimensions. A l'aide d'un moyen de contrôle quelconque, l'utilisateur choisit un point dans ce sous-espace. Une fois ce point choisi (et donc après avoir fixé les paramètres correspondants de la fonction), un nouveau graphe en trois dimensions est dessiné, permettant le choix de la valeur de trois nouvelles variables. Le processus se renouvelle jusqu'à épuisement des variables de la fonction. Le dernier graphe peut permettre de représenter des courbes, des surfaces ou un nuage de points en fonction des propriétés de f. A tout moment, l'utilisateur peut choisir de déplacer le point de référence dans chacun

[8]Si la dimension des données est paire, il suffit alors d'ajouter une dimension fictive aux données et de fixer la même valeur pour toutes les données.

des graphes 3D intermédiaires. L'avantage principal de cette représentation est de permettre une représentation exacte des données, cependant elle reste difficile à utiliser en raison de la complexité intrinsèque des fonctions f. Dans N-vision (Beshers and Feiner, 1993), les auteurs ont ajouté un système de règles (AutoVisual) pour faciliter l'utilisation de cette représentation, en assistant l'utilisateur quant au choix des variables. La figure 3.13 montre N-vision en action.

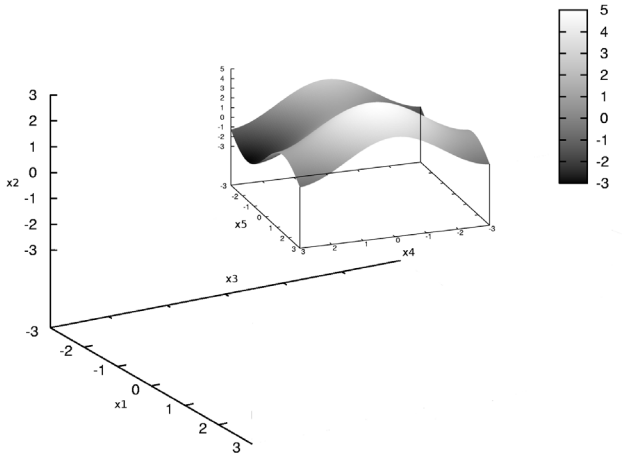


FIG. 3.13 – Exemple d'utilisation de la technique *Worlds within Worlds* pour la fonction $f(x_1, x_2, x_3, x_4, x_5) = x_1 * x_2 + x_3 * (cos(x_4) + sin(x_5))$.

3.5.8 Visualisations iconiques

Une visualisation iconique est une visualisation dans laquelle chaque donnée correspond à un élément visuel bien identifié et dans laquelle chaque élément visuel est obtenu à partir de primitives ou de couleurs paramétrées par les coordonnées du point. Ces éléments visuels peuvent être de différents types.

3.5.8.1 Les visages de Chernoff

Dans la visualisation à l'aide des visages de Chernoff (Chernoff, 1973), les éléments visuels sont des visages. Les primitives correspondent, par exemple, à l'ovale de la tête, la taille des yeux ou la longueur du nez, ... La figure 3.14 montre le résultat de cette technique.

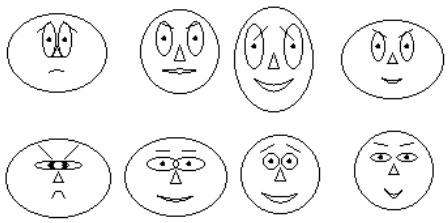


FIG. 3.14 – Visualisation d'un jeu de données fictif avec les visages de Chernoff. Cette figure a été construite grâce au logiciel *Chernoff faces in Java* (Wiseman, 1998).

3.5.8.2 *Star glyphs*

Dans la technique des *stars glyphs* (Siegel et al., 1972) (Chambers et al., 1983), les primitives correspondent à des segments d'origine le centre du star glyph. L'angle formé par deux segments adjacents est constant, tandis que la longueur de chaque segment dépend de la valeur prise par la coordonnée de chacune des données. Les extrémités des segments sont, en général, jointes de manière à fermer les figures. La figure 3.15 montre le résultat de cette technique.

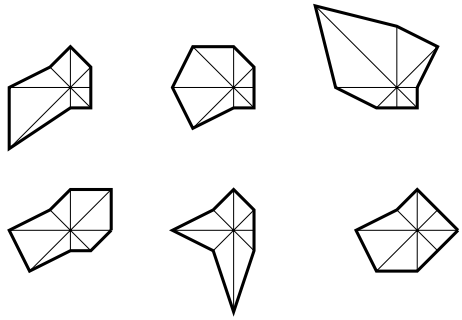


FIG. 3.15 – Visualisation d'un jeu de données fictif avec la technique des *stars glyphs*.

3.5.8.3 *Stick figures*

La technique des stick figures (Pickett and Grinstein, 1988) (Grinstein et al., 1989) est une approche similaire aux visages de Chernoff. Au lieu de paramétrer les différentes composantes d'un visage, la technique paramètre un enchaînement de segments (le nombre de segments et la structure pouvant varier selon les implémentations) : la longueur de chacun des segments et les angles formés sont des primitives graphiques. A la différence des visages de Chernoff, les sticks sont destinés à un affichage dense de points. Cet affichage compose alors une texture servant d'indicateur visuel sur les données. (Keim, 1997) présente un exemple d'application de cette technique.

3.5.8.4 Les courbes d'Andrews

Les courbes d'Andrews (Andrews, 1972) peuvent être considérées comme des visualisations iconiques. En effet, dans le premier cas, au lieu d'être un opérateur de positionnement spatial, l'opérateur de combinaison des primitives graphiques est une somme. Au lieu d'être des éléments graphiques, les primitives sont des fonctions. Pour cela, l'élément visuel associé à une donnée x de coordonnées $(x_1, x_2, \ldots, x_N)$ est la fonction définie par

$$f_x(t) \quad = \quad x_1 \frac{1}{\sqrt{2}} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \ldots$$

La représentation consiste donc à dessiner les différentes fonctions f_x en faisant varier t dans l'intervalle $]-\pi, \pi[$. De par ce mode de représentation, la visualisation peut être considérée comme une visualisation de la transformée de Fourier des données (Hoffmann and Grinstein, 2005).

3.5.9 Techniques à base de pixels

Les techniques à base de pixels sont des visualisations cousines des visualisations iconiques. En effet, dans ces visualisations, chaque composante de chaque donnée est représentée par un pixel de couleur. Mais, contrairement aux visualisations iconiques, les différents pixels

associés à une donnée ne sont pas regroupés dans l'espace. En lieu et place, les pixels correspondant à une même composante sont agencés ensemble dans la représentation. On trouve principalement deux techniques réalisant cette métaphore : les *circle segments* et VisDB.

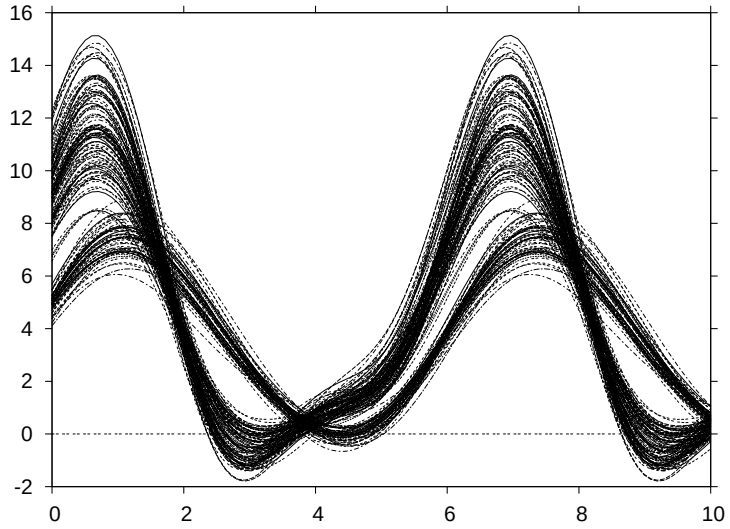


FIG. 3.16 – Visualisation de la base de données Iris (Fisher, 1936) selon la technique des courbes d'Andrews.

3.5.9.1 *Circle segments*

La technique de segment de cercle est une technique orientée pixel consistant à définir, pour chaque dimension des données, une portion angulaire d'un disque. Chacune des portions est de la même taille que les autres. On considère alors les données après les avoir triées dans un ordre donné (habituellement la valeur d'une dimension). Les valeurs des dimensions sont dessinées séparément suivant cet ordre dans chacune des portions. Pour une dimension donnée et par conséquent pour sa portion de disque, les valeurs prises sont représentées du centre du disque vers l'extérieur par remplissage successif d'un côté à l'autre et inversement (Ankerst et al., 1996) (Ankerst, 2001) (Ankerst, 2000). La couleur de chaque point indique la valeur prise par l'attribut. Le principe de remplissage est schématisé par la figure 3.17 (a). La figure 3.17 (b) montre le résultat de la technique sur un jeu de données importants.

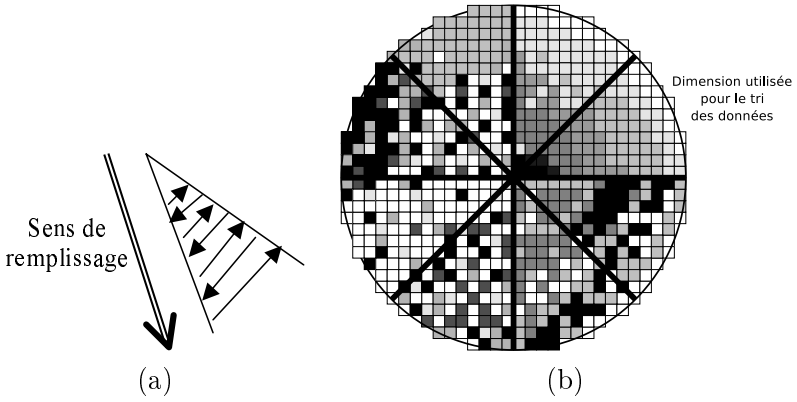


FIG. 3.17 – Visualisation avec les *circle segments* : (a) principe de remplissage d'une portion pour les *circle segments*, (b) exemple d'application de la technique de *circle segments* pour des données fictives en 8 dimensions.

3.5.9.2 VisDB

L'approche suivie par VisDB (Keim and Kriegel, 1994) (Ward and Keim, 1997) est similaire à celle suivie par les *circle segments*. Cependant, les pixels sont arrangés dans des rectangles juxtaposés les uns aux autres (cf. figure 3.19) au lieu d'être arrangés dans des portions de disques. La technique VisDB a pour objectif la visualisation du résultat d'une requête dans une base de données. Par conséquent, un attribut supplémentaire est ajouté aux données. La valeur de cet attribut mesure l'adéquation de la donnée à la requête.

L'ordre des pixels dans les rectangles est défini par la mesure d'adéquation de la donnée à la requête. Le positionnement est effectué en suivant une spirale à partir du centre du rectangle (cf. figure 3.18). Les colorations des pixels, aussi bien pour la mesure d'adéquation de la donnée à la requête que pour les dimensions des données, sont définies et ajustées par l'utilisateur. La visualisation simultanée de toutes les dimensions (cf. figure 3.19) permet de détecter les liaisons dans les données.

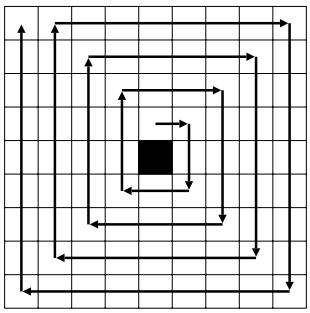


FIG. 3.18 – Visualisation du mode d'arrangement des pixels par la technique VisDB (Schéma adapté de (Keim and Kriegel, 1994)).

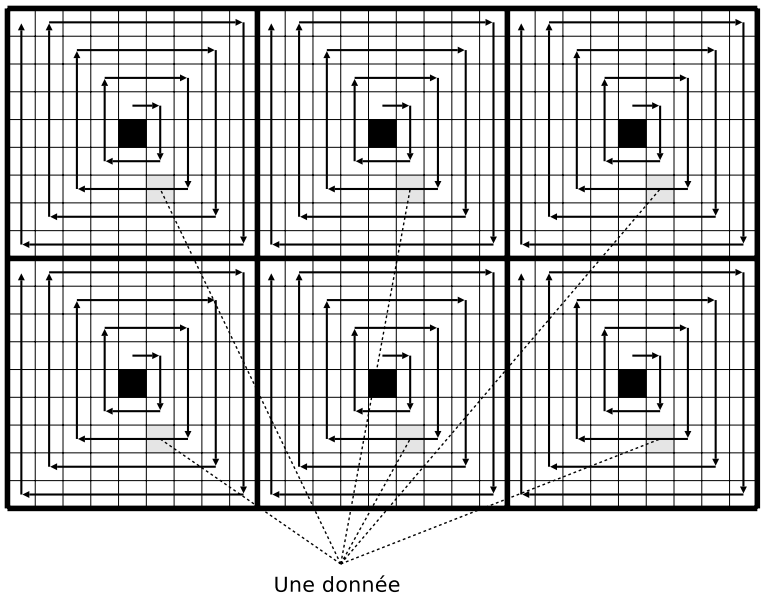


FIG. 3.19 – Visualisation simultanée de plusieurs dimensions par la technique VisDB (Schéma adapté de (Keim and Kriegel, 1994)).

3.6 Conclusion

Les techniques que nous venons d'évoquer ne représentent qu'une petite partie des techniques de visualisation d'information. Cependant elles regroupent, d'une façon ou d'une autre, l'ensemble des briques de base actuellement utilisées pour la construction de nouvelles représentations.

Lorsque l'on consulte la littérature du domaine, on se rend rapidement compte que, malgré le nombre assez impressionnant de visualisations disponibles dans la littérature, aucune brique de base véritablement originale n'est apparue depuis plusieurs années. En effet, les recherches actuelles ne font qu'essayer d'améliorer et d'agencer les briques existantes, en les appliquant à des problèmes spécifiques. De plus, les « nouvelles » visualisations ne sont réellement exploitées que par leurs créateurs. Une partie des recherches actuelles a donc plutôt tendance à s'orienter vers la conception d'un système logiciel complexe, intégrant de multiples visualisations, capables d'interagir les unes avec les autres, afin de les rendre plus accessibles aux utilisateurs.

Dans le cadre de cette thèse et notamment du chapitre 6, nous ne nous occuperons que de la définition d'une nouvelle technique de visualisation de dissimilarité, sans nous préoccuper de son intégration avec d'autres visualisations, que nous réservons pour des travaux futurs.

Deuxième partie

Contributions aux modèles de Markov cachés : apprentissage, nouveaux modèles, visualisation de dissimilarité et bibliothèque HMMTK

Chapitre 4

Métaheuristiques pour l'apprentissage de MMC

4.1 Introduction

La spécification d'un système d'intelligence artificielle utilisant des modèles de Markov cachés peut s'effectuer en trois phases distinctes, mais interagissantes entre elles (cf. figure 4.1). La première phase, que nous nommerons pré-traitement par la suite, consiste en l'ensemble des actions nécessaires à la transformation des données en séquences temporelles. La deuxième phase, dite d'apprentissage, consiste en la transformation de certaines des séquences construites en modèles de Markov cachés, grâce à un algorithme d'apprentissage, tel que ceux décrits aux chapitres 1 et 2. La dernière phase, dite de post-traitement, consiste en l'utilisation des MMC produits en deuxième phase et de séquences produites par la première phase pour effectuer le traitement. Les traitements pouvant être réalisés par un tel système sont très variés : classification, segmentation, analyse, décision, ...

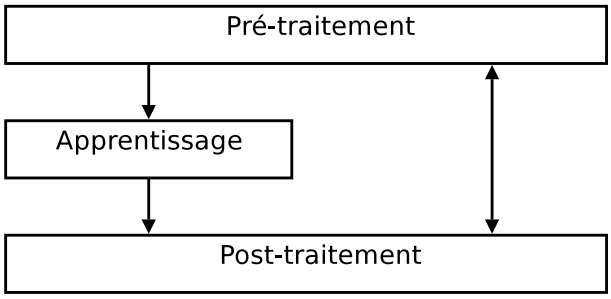


FIG. 4.1 – Phases de spécification d'un système d'intelligence artificielle utilisant des modèles de Markov cachés.

La phase d'apprentissage joue un rôle central au sein d'un tel système mais, en pratique, peu d'attention lui est accordée dans le cas des MMC. Dans de nombreuses applications, des modèles sous optimaux sont utilisés avec succès. Cependant, ces applications s'appuient sur des principes théoriques qui ne sont valables que lorsque les modèles sont optimaux. Par conséquent, il est communément admis que des modèles optimaux permettraient, du moins en théorie, d'améliorer les performances du système d'intelligence artificielle.

L'amélioration de l'apprentissage des MMC à l'aide de métaheuristiques à base de population est l'objet de ce chapitre. Nous commençons par présenter les différents espaces de solutions que nous avons considérés lors de nos recherches. Pour chacun d'eux, nous nous attacherons également à présenter, pour la suite de cet exposé, les propriétés mathématiques qui leurs sont associées. Nous poursuivrons par la description des métaheuristiques que nous avons

envisagées et développées. Ces algorithmes étant dérivés de métaheuristiques génériques, nous présenterons une analyse comparative des propriétés de chacune. Nous terminerons par une étude expérimentale de ces métaheuristiques.

4.2 Les espaces de solutions pour l'apprentissage de modèles de Markov cachés

L'objectif de cette section est de définir les espaces de solutions de MMC pour l'apprentissage. Au cours de ces travaux, nous avons considéré quatre types d'espaces de recherche pour un nombre N d'états cachés, un nombre M de symboles fixés et une séquence d'observations de longueur T : Λ, $\mathbb{S}^T$, Ω et Λ^*.

4.2.1 L'espace de solutions Λ

L'espace de solutions Λ est l'espace de solutions le plus couramment utilisé pour l'apprentissage de MMC. Il correspond à l'ensemble des triplets de matrices stochastiques (A, B, Π) définissant classiquement un modèle de Markov caché. Λ est donc isomorphe au produit cartésien $\mathbb{G}_N \times (\mathbb{G}_N)^N \times (\mathbb{G}_M)^N$ des espaces $\mathbb{G}_K$ de vecteurs stochastiques en dimension K. La propriété fondamentale de l'espace $\mathbb{G}_K$ est qu'il est convexe[1]. En généralisant sur le produit cartésien, on montre facilement que Λ est convexe.

4.2.2 L'espace de solutions $\mathbb{S}^T$

L'espace de solutions $\mathbb{S}^T$ correspond à l'ensemble des séquences d'états cachés de longueur T (*i.e.* la longueur de la ou des séquences d'observations, suivant le critère d'apprentissage utilisé). Les propriétés fondamentales de cet espace sont qu'il est discret et fini, de cardinal N^T. Un MMC étant défini par son triplet de matrices stochastiques, une solution de $\mathbb{S}^T$ ne peut pas être utilisée directement comme MMC. En lieu et place, on utilise l'algorithme d'apprentissage étiqueté avec ou sans lissage (cf. section 1.6.1) pour traduire cette séquence en MMC. Si l'on note O la ou les séquences d'observations suivant le critère considéré et $\gamma(Q)$ le modèle obtenu par l'algorithme d'apprentissage étiqueté à partir de O et la séquence d'états $Q \in \mathbb{S}^T$, alors l'ensemble $\gamma(\mathbb{S}^T) = \{\gamma(Q)/Q \in \mathbb{S}^T\}$ est un sous-ensemble fini de solutions de Λ.

Cet espace de solutions possède des avantages et des inconvénients. Le plus important de ses avantages est sa nature discrète, qui permet d'utiliser toute l'armada des métaheuristiques discrètes. En effet, il existe un très grand nombre de métaheuristiques pour des espaces de solutions discrètes, alors qu'il en existe relativement peu pour des espaces continus (Dréo et al., 2003). Le premier inconvénient est que, lorsque l'on utilise cet espace de recherche au lieu de Λ, on pose l'hypothèse que l'ensemble $\gamma(\mathbb{S}^T)$ contient suffisamment de modèles pour ne pas dégrader les possibilités de l'apprentissage. On peut aisément considérer cette hypothèse comme vraie si la séquence d'observations est suffisamment grande et si tous les symboles apparaissent plusieurs fois. Dans le cas contraire, il est possible que certaines probabilités ne puissent prendre qu'un nombre trop limité de valeurs[2]. Le deuxième inconvénient de cet espace de solutions est qu'il n'existe pas une séquence d'états cachés unique associée à un MMC.

[1] $\forall\ (x, y) \in (\mathbb{G}_K)^2, \alpha \in [0; 1],\ \sum_{i=1}^{K} \alpha x_i + (1-\alpha) y_i = 1,\ 0 \leq \alpha x_i + (1-\alpha) y_i \leq 1$.

[2] Les probabilités π_i ne prennent que deux valeurs, même avec un lissage, lorsqu'une seule séquence d'observations est utilisée avec le critère de maximum de vraisemblance car, au temps initial, le MMC est dans un et un seul état caché.

Pour le prouver, il suffit de considérer l'exemple suivant. Soit la séquence d'observations $O = (v_1v_2v_2v_2v_2)$ et deux séquences d'états $Q_1 = (s_1s_2s_3s_2s_2)$ et $Q_2 = (s_1s_2s_2s_3s_2)$. On a :

$$\gamma(Q_1) = \gamma(Q_2) = \lambda = (A, B, \Pi)$$

avec

$$\Pi = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 1 & 0 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

L'opérateur γ n'étant pas bijectif entre $\mathbb{S}^T$ et $\gamma(\mathbb{S}^T)$, il n'est pas possible d'utiliser d'opérateur d'optimisation locale tel que celui de Baum-Welch sans perdre l'intérêt de la représentation, ou perdre la cohérence de la méthode d'apprentissage.

4.2.3 L'espace de solutions Ω

Cet espace a été initialement défini afin de fournir une structure d'espace vectoriel pour l'apprentissage de MMC . A cet effet, on considère l'espace $\mathbb{G}_K$ des vecteurs stochastiques de dimension K. Soit $\mathbb{G}_K^* = \{x \in \mathbb{G}_K / \forall\ i = 1..K,\ x_i > 0\}$ le sous-espace des vecteurs stochastiques de dimension K dont aucune des composantes n'est nulle. Soit $r_K : \mathbb{R}^K \mapsto \mathbb{R}^K$ une fonction de régularisation définie sur $\mathbb{R}^K$ par :

$$r_K(x)_i = x_i - \max_{j=1..K} x_j$$

On définit $\Omega_K = r_K(\mathbb{R}^K) = \{x \in \mathbb{R}^K / r_K(x) = x\}$. La fonction de régularisation r_K est donc une projection sur Ω_K parallèlement à l'espace vectoriel $\mathbb{R} \cdot \mathbf{1}_K$ c'est-à-dire que $r_K(x)$ est l'unique élément de l'intersection $(x + \mathbb{R} \cdot \mathbf{1}_K) \cap \Omega_K$.

Les propriétés suivantes sont alors vérifiées :
- $r_K \circ r_K = r_K$
- pour tout $x \in \Omega_K$, $r_K(x) = x$
- pour tout $(x, y) \in (\mathbb{R}^K)^2$, $r_K(r_K(x) + r_K(y)) = r_K(x + y)$
- pour tout $x \in \mathbb{R}^K$, $c \in \mathbb{R}$, $r_K(c \cdot x) = r_K(c \cdot r_K(x))$
- pour tout $x \in \mathbb{R}^K$, $c \geq 0$, $r_K(c \cdot x) = c \cdot r_K(x)$
- pour tout $x \in \mathbb{R}^K$, $c \in \mathbb{R}$, $r_K(x + c \cdot \mathbf{1}_K) = r_K(x)$
- pour tout $x \in \mathbb{R}^K$, $r_K(-x) = r_K(-r_K(x))$

On définit les deux opérateurs symétriques suivants :
- $\oplus_K : \Omega_K \times \Omega_K \mapsto \Omega_K$ tel que, pour tout $(x, y) \in (\Omega_K)^2$, $x \oplus_K y = y \oplus_K x = r_K(x + y)$
- $\odot_K : \mathbb{R} \times \Omega_K \mapsto \Omega_K$ tel que, pour tout $x \in \Omega_K$ et $c \in \mathbb{R}$, $c \odot_K x = x \odot_K c = r_K(c \cdot x)$

Les propriétés suivantes sont alors vérifiées pour tout $(x, y, z) \in (\Omega_K)^3$ et $(a, b) \in \mathbb{R}^2$:
- $a \odot_K (x \oplus_K y) = (a \odot_K x) \oplus_K (a \odot_K y)$
- $(a + b) \odot_K x = (a \odot_K x) \oplus_K (b \odot_K x)$
- $a \odot_K (b \odot_K x) = (a \cdot_K b) \odot_K x$
- $(x \oplus_K y) \oplus_K z = x \oplus_K (y \oplus_K z)$
- $1 \odot_K x = x$

L'ensemble de ces propriétés montre que $(\Omega_K, \oplus_K, \odot_K)$ est un espace vectoriel. On définit $\ominus_K : \Omega_K \times \Omega_K \mapsto \Omega_K$ tel que, pour tout $(x, y) \in (\Omega_K)^2$,

$$x \ominus_K y = y \ominus_K x = r_K(x - y) = x \oplus (-1 \odot_K y)$$

On définit les opérateurs ϕ et ψ permettant de transformer les éléments de $\mathbb{G}_K^*$ en les éléments de Ω_K et réciproquement par :
- $\psi_K : \mathbb{G}_K^* \mapsto \Omega_K$ tel que pour tout $x \in \mathbb{G}_K^*$, $\psi_K(x)_i = \ln x_i - \max_{j=1..K} \ln x_j$
- $\phi_K : \Omega_K \mapsto \mathbb{G}_K^*$ tel que pour tout $x \in \Omega_K$, $\phi_K(x)_i = \frac{\exp x_i}{\sum_{j=1..K} \exp x_j}$.

$$\mathbb{G}_K^* \underset{\phi_K}{\overset{\psi_K}{\underset{\longleftarrow}{\longrightarrow}}} \Omega_K$$

Les deux propriétés suivantes sont vérifiées :
- pour tout $x \in \mathbb{G}_K^*$, $\phi_K(\psi_K(x)) = x$
- pour tout $x \in \Omega_K$ et $c \in \mathbb{R}$, $\phi_K(x + c \cdot \mathbf{1}_K) = \phi_K(x)$

L'espace $(\Omega_K, \oplus_K, \odot_K)$ peut être équipé de deux normes :
- $||x||_2$ défini pour tout $x \in \Omega_K$ par ${||x||_2}^2 = \sum_{i=1..K} x_i^2$.
- $||x||_{\max}$ défini pour tout $x \in \Omega_K$ par $||x||_{\max} = \max_{i=1..K} |x_i|$

Ces deux normes vérifient, pour tout $x \in \Omega_K$ et $c \in \mathbb{R}$, la propriété $||c \cdot x|| = |c| \cdot ||x||$ permettant en conséquence de normer les vecteurs de l'espace.

On définit $\Omega = \Omega_N \times (\Omega_N)^N \times (\Omega_M)^N$ et $\Lambda^* = \mathbb{G}_N \times (\mathbb{G}_N)^N \times (\mathbb{G}_M)^N$. En généralisant les opérateurs $\oplus_K$, $\odot_K$, $\ominus_K$, ψ_K, ϕ_K aux produits cartésiens Ω et Λ^*, on obtient les propriétés suivantes, après suppression de l'indice K :
- $\psi(\Lambda^*) = \Omega$
- $\phi(\Omega) = \Lambda^*$
- $(\Omega, \oplus, \odot)$ est un espace vectoriel

La généralisation des deux normes précédentes peut s'effectuer de différentes façons. Nous avons retenu les formes suivantes :
- $||x||_2$ défini pour tout $x \in \Omega$ par ${||x||_2}^2 = \sum_{i=1..K} x_i^2$
- $||x||_{\max}$ défini, pour tout élément de Ω, par le maximum des normes $||.||_{\max}$ sur chacun des sous espaces Ω_K du produit cartésien.

4.2.4 L'espace Λ^*

Nous venons de montrer qu'il était possible de plonger l'espace des MMC Λ^* dans un espace Ω possédant la structure d'espace vectoriel. Dans cette section, nous montrons comment une structure d'espace vectoriel similaire peut directement être définie sur Λ^*.

En empruntant une démarche similaire à la section précédente, nous établissons une structure vectorielle sur $\mathbb{G}_K^*$, que nous généralisons à Λ^*. On définit les deux opérateurs suivants :
- $\boxplus_K : \mathbb{G}_K^* \times \mathbb{G}_K^* \mapsto \mathbb{G}_K^*$ tel que, pour tout $(x, y) \in (\mathbb{G}_K^*)^2$, $x \boxplus_K y = y \boxplus_K x$ avec

$$(x \boxplus_K y)_i = \frac{x_i \cdot y_i}{\sum_{j=1}^{K} x_j \cdot y_j}$$

- $\boxdot_K : \mathbb{G}_K^* \times \mathbb{G}_K^* \mapsto \mathbb{G}_K^*$ tel que, pour tout $x \in \mathbb{G}_K^*$ et $c \in \mathbb{R}$, $c \boxdot_K x = x \boxdot_K c$ avec

$$(c \boxdot_K x)_i = \frac{x_i^c}{\sum_{j=1}^{K} x_j^c}$$

Le vecteur $\mathbf{0}$ de l'opérateur $\boxplus_K$ est le vecteur stochastique en dimension K dont toutes les composantes sont égales à $1/K$, c'est-à-dire que $\mathbf{0}$ est le vecteur équiprobable en K dimensions. Il est alors aisé de montrer que $(\mathbb{G}_K^*, \boxplus, \boxdot)$ est un espace vectoriel.

Pour tout $(x, y) \in (\mathbb{G}_K^*)^2$, on définit $(x', y') \in (\Omega_K)^2$ tels que $x' = \psi(x)$ et $y' = \psi(y)$. Pour tout $c \in \mathbb{R}$, on montre facilement les propriétés suivantes :

- $x \boxplus y = \phi(\psi(x) \oplus \psi(y))$
- $c \boxdot x = \phi(c \odot \psi(y))$
- $\phi(x' \oplus y') = \phi(x') \boxplus \phi(y')$
- $\phi(c \odot x') = c \boxdot \phi(x')$

Ces propriétés montrent que les structures vectorielles définies sur Ω_K et $\mathbb{G}_K^*$ sont équivalentes. Il est donc possible de munir $\mathbb{G}_K^*$ d'une structure vectorielle plus simplement qu'en utilisant ψ et ϕ pour effectuer les calculs dans Ω_K.

Tout comme précédemment, les opérateurs $\boxplus_K$ et $\boxdot_K$ se généralisent facilement au produit cartésien $\Lambda^* = \mathbb{G}_N^* \times (\mathbb{G}_N^*)^N \times (\mathbb{G}_M^*)^N$.

$(\Lambda^*, \boxplus, \boxdot)$ définit un espace vectoriel.

Il est possible de transposer les deux normes de la section précédente sur $\mathbb{G}_K^*$:

- $||x||_{\Omega_K}$ défini pour tout $x \in \mathbb{G}_K^*$ par $||x||_{\Omega_K} = ||\psi_K(x)||_2$. On vérifie aisément que la propriété $||\psi_K(c \boxdot x)||_2 = ||c \odot \psi_K(x)||_2 = |c| \cdot ||\psi_K(x)||_2$ est vérifiée.
- $||x|| = \left|\ln \frac{\min_{i=1..K} x_i}{\max_{i=1..K} x_i}\right|$. On montre alors que $||x|| = ||\psi_K(x)||_{\max}$.

La généralisation des deux normes précédentes peut s'effectuer de différentes façons. Nous avons retenu les formes suivantes correspondant à celles de la section précédente :

- $||x||_{\Omega_K}$ défini pour tout $x \in \Lambda^*$ par $||x|| = ||\psi(x)||_2$
- $||x||$ défini pour tout élément de Λ^* par le maximum des normes $||.||$ sur chacun des sous espaces $\mathbb{G}_K^*$ du produit cartésien.

4.2.5 Remarques sur les espaces de recherche

Lors de la construction des espaces de recherche, nous avons distingué les ensembles Λ et Λ^*. Cependant, dans la pratique, en raison des problèmes de précision numérique, l'espace Λ est toujours utilisé en lieu et place de Λ^*. Il est donc nécessaire d'adapter certains opérateurs problématiques de manière à ce qu'ils puissent manipuler des probabilités nulles. Une approche classique, que nous avons adoptée, consiste à seuiller la valeur des probabilités, de manière à ce qu'elles soient toujours supérieures à 0.000001.

Les espaces $(\Lambda^*, \boxplus, \boxdot)$ et $(\Omega, \oplus, \odot)$ étant équivalents, nous avons utilisé une forme ou l'autre de façon équivalente lors de nos travaux.

On peut également remarquer que, lors d'un apprentissage dans lequel les états cachés n'ont pas de rôle particulier défini *a priori*, un MMC équivalent à un MMC λ_1 s'obtient en renumérotant les états. Par conséquent, il n'existe pas un optimum, mais plusieurs, et l'espace de recherche possède de nombreuses symétries, quel que soit le critère d'apprentissage, si les rôles ne sont pas définis. Cette remarque pourrait permettre de réduire l'espace des solutions. Cependant, cette approche ne semble pas avoir été utilisée à notre connaissance. Nous ne l'avons pas utilisée dans la suite de ces travaux, afin de ne pas réduire les propriétés mathématiques de l'espace des solutions considéré.

Le tableau 4.1 synthétise l'utilisation des espaces de recherche dans les algorithmes d'apprentissage de MMC existants décrits aux chapitres 1 et 2. Le lecteur notera que l'algorithme de *segmental k-means* passe alternativement d'un espace à l'autre.

4.3 Les métaheuristiques pour l'apprentissage de MMC

Dans cette section, nous nous attachons à la description des métaheuristiques que nous avons considérées ou créées pour l'apprentissage de modèles de Markov cachés.

TAB. 4.1 – Positionnement des algorithmes existants d'apprentissage de MMC par rapport aux espaces de recherche.

Algorithme \ Espace	Λ	$\mathbb{S}^T$
Apprentissage étiqueté		X
Baum-Welch	(Baum and Eagon, 1967)	
Information mutuelle	(Rabiner, 1989) (Schluter et al., 1997) (Vertanen, 2004)	
Segmental k-means	(Juang and Rabiner, 1990)	
Recuit simulé	(Paul, 1985)	(Hamam and Al Ani, 1996)
Recherche tabou	(Chen et al., 2004)	
Algorithme génétique	(Slimane et al., 1999), (Brouard, 1999) (Thomsen, 2002)	
Apprentissage incrémental à base de population	(Maxwell and Anderson, 1999)	
API (fourmis artificielles)	(Monmarché, 2000)	
Optimisation par essaim particulaire	(Rasmussen and T. Krink, 2003)	

4.3.1 La métaheuristique AG

La première métaheuristique, dénommée AG, que nous avons considérée est à la fois un cas particulier de l'algorithme GHOSP (cf. section 2.5.4) et une extension. Il s'agit d'un cas particulier, dans le sens où nous reprenons l'algorithme GHOSP, mais en considérant une population de MMC ayant tous le même nombre d'états cachés. Il s'agit d'une extension, car nous ajoutons deux paramètres binaires à l'algorithme : *MuterParents* et *OptimiserParents* indiquant respectivement si l'opérateur de mutation et l'opérateur d'optimisation sont appliqués à la population parent. L'algorithme AG, qui en est issu, est donné par 4.1. Pour mémoire, nous rappelons que l'opérateur de mutation consiste à modifier chaque coefficient de chaque modèle avec la probabilité p_{mut} et que l'opérateur d'optimisation consiste à appliquer $\mathcal{N}_{\text{BW}}$ itérations de l'algorithme de Baum-Welch à chaque individu.

4.3.2 La métaheuristique API

La deuxième métaheuristique, dénommée API, que nous avons considérée est celle décrite à la section 2.7.2. L'apprentissage de MMC a été abordé par N. Monmarché (Monmarché, 2000) ; cependant, il n'a été étudié que pour des jeux de tests simples et sans évaluer l'impact précis des paramètres de la méthode. Nous avons donc cherché à pallier ces problèmes, afin de pouvoir la considérer ou non comme un outil fiable, dans le cas des MMC.

Pour mémoire, nous rappelons les paramètres de l'algorithme de la section 2.7.2 :

- $\mathcal{T}_{\text{max}}$: le nombre d'itérations de l'algorithme. Cette valeur correspond au nombre de sorties du nid effectuées par chacune des fourmis.
- $\mathcal{T}_{\text{Déplacement}}$: le nombre d'itérations de l'algorithme entre deux déplacements du nid.
- $\mathcal{N}$: le nombre de fourmis
- M_{max} : la taille de la mémoire d'une fourmi
- e_{max} : la patience d'une fourmi pour un site de chasse
- A_i^{Local} et A_i^{Site} : les amplitudes locales et de site pour l'exploration de solutions

Algorithme 4.1: Algorithme génétique pour l'apprentissage de MMC

Initialiser la population de $\mathcal{N}$ individus : P_0
Évaluer les individus de la population P_0
$t = 0$
Répéter
  Sélectionner les $\mathcal{N}_{\text{parent}}$ meilleurs individus pour la reproduction : $P_t^{\text{parent}} \subseteq P_t$
  Croiser les individus de P_t^{parent} pour obtenir P_t^{enfant}
  Si *MuterParents* **Alors**
    Muter les individus de P_t^{parent}
  Fin Si
  Si *OptimiserParents* **Alors**
    Optimiser les individus de P_t^{parent}
  Fin Si
  Muter les individus de P_t^{enfant}
  Optimiser les individus de P_t^{enfant}
  $P_{t+1} = P_t^{\text{parent}} \bigcup P_t^{\text{enfant}}$
  Évaluer les individus de la population P_{t+1}
  $t = t + 1$
Tant que $t < \mathcal{T}_{\max}$

4.3.3 La métaheuristique OEPDistance

La première métaheuristique que nous avons réalisée est une adaptation de l'optimisation par essaim particulaire à l'apprentissage de MMC (Aupetit et al., 2005a). L'OEP, de par sa nature et ses origines, se destine à des espaces de solutions dans lesquels la notion de « vecteur » a un sens (par exemple pour la vitesse).

L'adaptation de l'OEP pour cette tâche, réalisée par Rasmussen et Krink dans (Rasmussen and T. Krink, 2003), se contente de faire évoluer les particules dans un espace réel et de ne considérer l'espace des MMC Λ qu'au moment de l'évaluation. A notre avis, cette approche, bien que simple, fait que, indépendamment de la renumérotation des états (cf. section 4.2.5), plusieurs solutions de l'espace réel correspondent à un unique MMC de Λ. Pour le montrer, il suffit de prendre un vecteur $x \in \mathbb{R}^K$ et de considérer la transformation[3] δ de $\mathbb{R}^K$ vers $\mathbb{G}_K$ consistant à prendre la valeur absolue des composantes et à normaliser la somme à l'unité : $\delta(x)_i = \frac{|x_i|}{\sum_{j=1}^{K} |x_k|}$. Avec une telle transformation, on remarque que quel que soit $c \in \mathbb{R}$, les vecteurs de la forme $c \cdot x$ ont tous la même image. Par conséquent, ce type de représentation a pour effet de créer un nombre infini de « symétries » dans l'espace des solutions et donc de potentiellement complexifier la recherche d'un bon modèle. Dans (Rasmussen and T. Krink, 2003), les auteurs arrivent à obtenir de meilleurs résultats que le seul algorithme de Baum-Welch en utilisant une structure de MMC très contrainte, et grâce à l'ajout de bonnes solutions parmi les particules, garantissant une performance au moins égale aux algorithmes permettant d'obtenir ces bonnes solutions.

Dans l'adaptation que nous proposons, nous avons adopté une approche différente, ayant pour principal avantage de travailler sur un espace vectoriel ne créant pas de multiples « symétries ». Dans ce but, nous avons construit l'espace de solutions Ω et nous l'avons muni de la structure vectorielle $(\Omega, \oplus, \odot)$ (cf. section 4.2.3 pour plus de détails). A l'aide des fonctions de transformation ψ et ϕ, nous avons montré que les espaces Λ^* et Ω sont isomorphes et en

[3] Dans (Rasmussen and T. Krink, 2003), les auteurs ne précisent pas quelle transformation est utilisée, mais la transformation évoquée nous paraît un choix raisonnable.

conséquence il n'y a pas plus de « symétries » dans Ω que dans Λ^*. La recherche de bonnes solutions dans Λ^* est équivalente à la recherche de solutions dans Ω.

$$\Lambda^* \underset{\phi}{\overset{\psi}{\rightleftarrows}} \Omega$$

Dans la section 4.2.4, nous avons montré que les espaces $(\Omega, \oplus, \odot)$ et $(\Lambda^*, \boxplus, \boxdot)$ sont équivalents et que leurs opérateurs sont également équivalents. Par conséquent, bien que la description suivante adopte le formalisme de l'espace Ω, nous aurions pu tout autant utiliser le formalisme lié à Λ^*.

L'adaptation que nous proposons intègre deux particularités supplémentaires : une étape d'optimisation locale et des voisinages spatiaux. L'étape d'optimisation locale est effectuée au moyen de l'algorithme de Baum-Welch selon l'équation suivante :

$$x_i(t) = \text{Optimisation}(x_i(t-1) + v_i(t-1))$$

La particule subit le déplacement habituel, en ajoutant à sa position le vecteur vitesse associé. La position résultante est alors optimisée grâce à $\mathcal{N}_{\text{BW}}$ itérations de l'algorithme de Baum-Welch. Pour conserver la cohérence du déplacement lors de la mise à jour de la vitesse, la vitesse précédente est également mise à jour de manière à refléter le déplacement réalisé par l'équation :

$$v_i(t-1) = x_i(t) - x_i(t-1)$$

Cette mise à jour permet à l'optimisation locale de guider la recherche, plutôt que de s'y opposer en provoquant des mouvements contraires (la direction de $v_i(t)$ s'opposant à celle due à l'optimisation locale). Pour cette première adaptation de l'OEP, nous avons utilisé des voisinages spatiaux. Pour cela, le voisinage $V_i(t)$ d'une particule i à l'instant t est composé des V particules qui lui sont les plus proches dans l'espace, au sens de la mesure d. L'algorithme OEPDistance est donné par l'algorithme 4.2.

4.3.4 La métaheuristique OEPSocial

Nous avons conçu une deuxième adaptation de l'OEP pour l'apprentissage de MMC. Cet algorithme, dénommé OEPSocial, reprend presque totalement la structure de l'algorithme OEPDistance, à l'exception du voisinage des particules, qui devient fixe et avec une structure circulaire. Lorsque la taille du voisinage est V, le voisinage $V_i(t)$ de la particule i à l'instant t est constant et égal à V_i. V_i est composé des $V/2$ particules la précédant et des $V/2$ particules lui succédant, lorsque les particules sont disposées sur un cercle. La figure 4.2 montre la composition du voisinage de taille 2 des particules 1 et 5.

4.3.5 La métaheuristique AGDiscret

Nous avons conçu un algorithme génétique, dénommé AGDiscret, qui effectue la recherche de MMC en utilisant l'espace $\mathbb{S}^T$. Cet algorithme est similaire à l'algorithme AG décrit plus haut. Pour cet algorithme, les individus sont représentés par une séquence d'états cachés étiquetant la ou les séquences d'observations à apprendre. Le MMC correspondant est alors obtenu grâce à l'algorithme d'apprentissage étiqueté. L'opérateur de sélection utilisé est l'opérateur de sélection élitiste. L'opérateur de croisement correspond à l'opérateur classique à un (1) point (1X) des algorithmes génétiques. L'opérateur de mutation consiste à modifier chaque élément de la séquence d'états avec la probabilité p_{mut}. La modification d'un élément de la séquence consiste à engendrer uniformément un état dans $\mathbb{S}$.

Comme nous l'avons dit précédemment, en raison de la non réversibilité de l'algorithme d'apprentissage étiqueté, il n'est pas possible d'utiliser un algorithme d'optimisation locale tel

Algorithme 4.2: L'algorithme OEPDistance

Pour $i = 1$ **à** $\mathcal{N}$ **Faire**
  $\mathbf{x}_i(0) = \mathcal{U}(\psi(\Lambda))$
  $\mathbf{v}_i(0) = \mathcal{U}(\psi(\Lambda))$
  $\mathbf{x}_i^+(0) = \mathbf{x}_i(0)$
Fin Pour
Pour $t = 1$ **à** $\mathcal{T}_{\max}$ **Faire**
  Pour $i = 1$ **à** $\mathcal{N}$ **Faire**
    // Déplacement
    $\mathbf{x}' = \mathbf{x}_i(t-1) \oplus \mathbf{v}_i(t-1)$
    $\mathbf{x}_i(t) = \psi(BW(\phi(\mathbf{x}')))$
    $\mathbf{v}_i(t) = \mathbf{x}_i(t) \ominus \mathbf{x}_i(t-1)$
    // Mise à jour de la mémoire
    Si $P(O|\phi(\mathbf{x}_i(t))) > P(O|\phi(\mathbf{x}_i^+(t-1)))$ **Alors**
      $\mathbf{x}_i^+(t) = \mathbf{x}_i(t)$
    Sinon
      $\mathbf{x}_i^+(t) = \mathbf{x}_i^+(t-1)$
    Fin Si
  Fin Pour
  Pour $i = 1$ **à** $\mathcal{N}$ **Faire**
    // Calcul du voisinage
    $\mathcal{V}_i(t) = \emptyset$
    Tant que $|\mathcal{V}_i(t)| < \mathcal{V}$ **Faire**
      $\mathcal{V}_i(t) = \mathcal{V}_i(t) \cup \arg \min_{k \notin \mathcal{V}_i(t)} d(\mathbf{x}_i(t), \mathbf{x}_k^+(t))$
    Fin Tant que
    // Calcul de l'influence dans le voisinage
    $\hat{\mathbf{x}}_i(t) = \mathbf{x}_j(t)^+$ avec $j = \arg \max_{k \in \mathcal{V}_i(t)} P(V = O|\phi(\mathbf{x}_k^+(t)))$
    // Mise à jour de la vitesse
    $\mathbf{v}_i(t) = \omega \odot \mathbf{v}_i(t-1)$
      $\oplus\, [c_1 \cdot \mathcal{U}([0,1])] \odot (\mathbf{x}_i^+(t) \ominus \mathbf{x}_i(t))$
      $\oplus\, [c_2 \cdot \mathcal{U}([0,1])] \odot (\hat{\mathbf{x}}_i(t) \ominus \mathbf{x}_i(t))$
  Fin Pour
Fin Pour

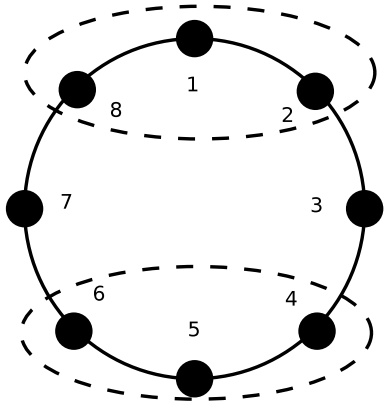


FIG. 4.2 – Exemple de voisinage circulaire.

que l'algorithme de Baum-Welch sans trahir les principes des algorithmes génétiques. Pour préserver ces principes, il serait nécessaire d'effectuer l'optimisation locale dans l'espace $\mathbb{S}^T$, par exemple par des essais de solutions dans le voisinage (comme le feraient une recherche tabou ou un recuit simulé). Cependant, même si cette approche paraît séduisante au premier abord, elle ne l'est pas car elle nécessite d'évaluer la fonction objectif pour chacune des solutions. Or, chaque évaluation nécessite l'utilisation de l'algorithme d'apprentissage étiqueté et, dans le cas le plus simple, de l'algorithme *Forward*. On peut également remarquer que plus la séquence est longue, moins la modification d'un état de la séquence n'a d'impact sur les valeurs des matrices du modèle correspondant. Or, il n'existe pas, à notre connaissance, d'algorithme en complexité inférieure à celle de *Forward* capable de calculer l'impact d'une modification sur la probabilité finale. Un tel coût étant prohibitif, nous n'avons donc pas inclus d'optimisation locale des solutions dans AGDiscret. Comme pour l'algorithme AG, nous autorisons la population parente à subir l'opérateur de mutation en fonction de la valeur du paramètre *MuterParents* et l'opérateur d'optimisation locale en fonction de la valeur du paramètre *OptimiserParents*.

4.3.6 La métaheuristique APIDiscret

De la même façon que pour les algorithmes génétiques, nous avons cherché à adapter l'algorithme API à l'espace de solutions $\mathbb{S}^T$. L'adaptation de cet algorithme, dénommé APIDiscret, à cet espace de solutions s'effectue facilement. Pour définir cet algorithme, il est nécessaire de spécifier deux opérateurs : l'opérateur de génération de la position initiale du nid et l'opérateur d'exploration locale. Le premier opérateur consiste tout simplement en la génération uniforme d'une solution dans $\mathbb{S}^T$. L'opérateur d'exploration locale autour d'une solution x pour une amplitude $A \in [0;1]$ consiste à modifier $L = \min\{A \cdot T \cdot \mathcal{U}([0;1]), 1\}$ états de la séquence x. Les états à modifier sont choisis aléatoirement dans la séquence et leurs valeurs sont engendrées uniformément dans $\mathbb{S}$. Pour des raisons identiques à celles évoquées pour AGDiscret, nous n'avons pas considéré d'opérateur d'optimisation locale au sein de l'opérateur d'exploration locale.

4.3.7 La métaheuristique API*

La dernière métaheuristique, dénommé API*, que nous avons considérée est une adaptation de l'algorithme API (cf. section 2.7.2) sur $(\Lambda^*, \boxplus, \boxdot)$ (Aupetit et al., 2005d). API* reprend la même structure que l'adaptation de API sur Λ de la section 4.3.2. Les opérateurs de génération de la position initiale du nid et d'exploration locale sont redéfinis. L'opérateur de génération de la position initiale du nid consiste tout simplement à générer une solution uniformément dans Λ^*.

L'opérateur d'exploration locale autour d'une solution x pour une amplitude $A \in [0;\mu]$ ($\mu > 0$) a besoin d'être défini avec précaution. Supposons que l'amplitude A soit fixée à une valeur quelconque. Soit $y = \frac{A}{||x||} \boxdot x$ pour $x \in \mathbb{G}_K$. Deux cas extrêmes peuvent se produire. y est presque égal à x si les coefficients de x sont suffisament similaires et inversement y tend vers le vecteur de probabilités nul comportant un seul coefficient à 1. Ces deux cas extrêmes dépendent de deux facteurs : l'amplitude A et les coefficients de $\frac{1}{||x||} \boxdot x$.

Soit $x = (x_1, x_2)' \in \mathbb{G}_2$. Soit $f(x) = x_1$. On a $\phi_2(x) = (f(x), 1 - f(x))'$. L'effet de l'amplitude $A \in [0, \mu]$ sur le vecteur x s'évalue donc par la largeur $\Delta(\mu, x)$ de l'intervalle de valeur occupé par $f(x)$, c'est-à-dire

$$\Delta(\mu, x) = |f(0 \boxdot x) - f(M \boxdot x)| = \left| \frac{1}{2} - \frac{x_1^\mu}{x_1^\mu + (1 - x_1)^\mu} \right|$$

La figure 4.3 présente l'évolution de l'amplitude $\Delta(\mu, x)$ de l'intervalle en fonction de la valeur de $x = (x_1, 1 - x_1)'$ et de l'amplitude maximum μ. Comme nous pouvons le voir, la largeur de l'intervalle peut varier de 0 à 0.5. Par conséquent, cette approche n'est pas satisfaisante.

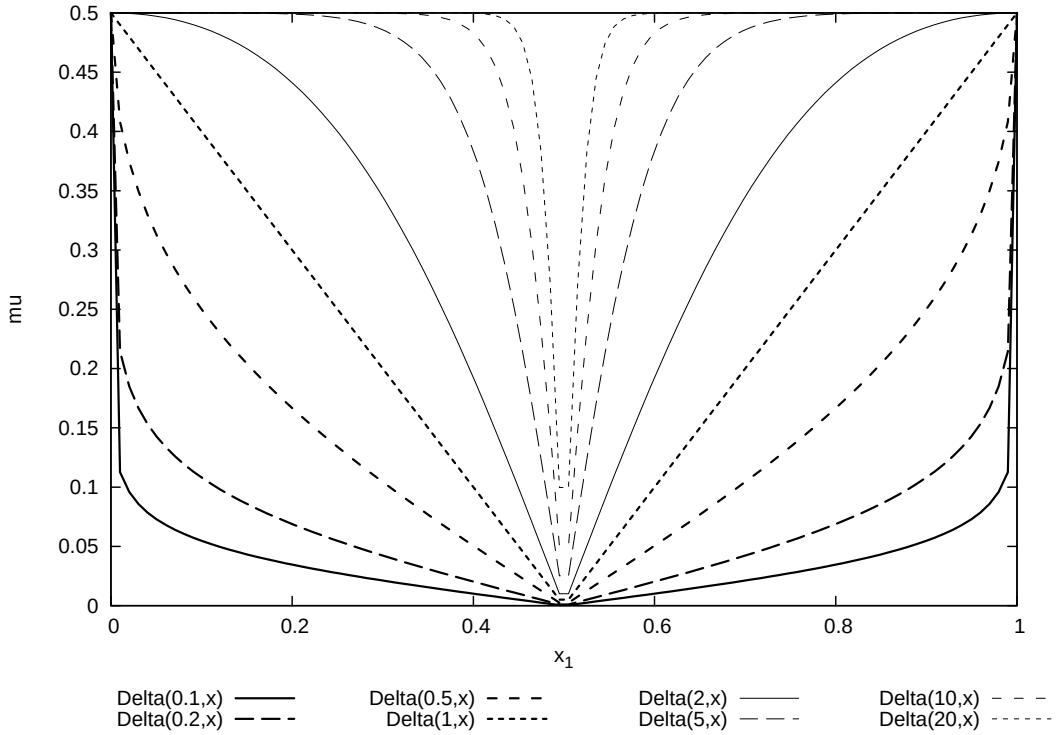


FIG. 4.3 – Evolution de l'amplitude $\Delta(M, x)$ de l'intervalle en fonction de la valeur de $x = (x_1, 1 - x_1)'$ et de l'amplitude maximum M.

Un moyen utilisable pour réduire ces cas extrêmes consiste à considérer la norme $||x|| = \left|\ln \frac{\min_{i=1..K} x_i}{\max_{i=1..K} x_i}\right|$ et à remarquer que

$$||x|| = c \quad \Longleftrightarrow \quad \min_{i=1..K} x_i = e^{-c} \max_{i=1..K} x_i$$

Or, $e^{-c} \in]0;1]$. Si c est choisi uniformément dans $[0;M]$, e^{-c} ne l'est pas dans $]0;1]$. Une meilleure façon de choisir c tout en s'assurant que le rapport $\frac{\min_{i=1..K} x_i}{\max_{i=1..K} x_i}$ est choisi de manière uniforme, consiste à choisir e^{-c} de manière uniforme. En prenant $M = 1$ et $y = \mathcal{U}(\Lambda^*)$, l'opérateur $\mathcal{O}_{\text{Explo}}(A, x)$ s'exprime par

$$\mathcal{O}_{\text{Explo}}(A, x) = x \boxplus \left(\frac{-\ln\left(\mathcal{U}(]A;1[)\right)}{||y||} \boxdot y \right)$$

Les paramètres de API* sont alors identiques à ceux de API.

4.3.8 Positionnement des algorithmes par rapport aux espaces de solutions

Afin de faciliter l'analyse de tous ces algorithmes, nous proposons de les situer par rapport aux espaces de solutions. Le tableau résultant est donné par la table 4.2.

TAB. 4.2 – Positionnement des algorithmes par rapport aux espaces de solutions.

Algorithme \ Espace	Λ	$\mathbb{S}^T$	Λ^*, Ω
AG	X	X	
API	X	X	X
OEP Distance/Social			X

4.4 Comparaison théorique des métaheuristiques

Les algorithmes décrits ci-dessus sont fondés sur des métaheuristiques génériques à la fois similaires et différentes dans leur stratégie d'exploration de l'espace des solutions. L'objectif de cette section est de mettre en valeur les similitudes et les différences de ces métaheuristiques génériques, afin de mieux comprendre le fonctionnement des adaptations réalisées.

Ces métaheuristiques sont toutes composées d'une population d'agents interagissant les uns avec les autres afin de trouver la meilleure solution. L'interaction des agents entre eux s'effectue avec des moyens, des moments et des individus différents. L'interaction au sein de l'algorithme génétique s'effectue via le croisement des meilleures solutions par l'échange et le partage de gènes, à chacune des itérations de l'algorithme. Par conséquent, en présence d'un codage des gènes adaptés, chaque itération permet le transfert des bonnes propriétés des solutions aux individus enfants. Dans l'optimisation par essaim particulaire, l'interaction est effectuée à chaque itération via la mise à jour des vecteurs vitesses. Cependant, la transmission directe des bonnes propriétés des particules, telles que la position, ne se fait pas. En lieu et place, la direction vers les meilleures particules du voisinage est opérée. L'algorithme API adopte une stratégie totalement différente : l'interaction ne s'effectue que périodiquement toutes les $\mathcal{T}_{\text{Déplacement}}$ itérations lorsque le nid est déplacé. Cette interaction peut être qualifiée de directe, dans le sens où le nid est une solution qui est transmise à chacune des fourmis. Chacune de ces approches possède ses avantages et ses inconvénients. Interagir à chacune des itérations, comme pour l'AG et l'OEP, permet de converger plus rapidement vers de bonnes solutions, mais cette fréquence d'intéraction augmente le risque que tous les agents se déplacent dans la même direction, réduisant de ce fait la diversité de l'exploration. De plus, si plusieurs bonnes solutions comparables du point de vue du critère d'optimisation sont rencontrées par les agents, il est fortement possible que des pôles d'attractions antagonistes provoquent des oscillations des agents entre ces pôles et que de nombreux efforts d'explorations soient perdus. Cependant, cette répartition des efforts sur plusieurs pôles est parfois nécessaire pour garantir une convergence vers une solution optimale ou presque optimale. D'un autre côté, des interactions rares, comme dans le cas de API, peuvent réduire l'efficacité de la recherche par une recherche en aveugle sans connaissance sur le reste de l'espace des solutions. Cette rareté des interactions peut également être un avantage, car elle permet de réduire le nombre d'itérations nécessaires pour prendre une décision parmi des pôles antagonistes : la décision étant prise au moment où le nid est déplacé. Ce type de décision trop expéditive peut être préjudiciable dans certains cas. Dans tous les cas de figure, le moment où s'effectue la prise de décision (interaction) peut être un avantage ou un inconvénient, en fonction de la nature de l'espace des solutions : hésiter entre plusieurs solutions est plus sûr pour trouver un optimum mais cela réduit la vitesse de convergence de l'algorithme. La figure 4.4 synthétise le mode et la fréquence de l'interaction dans l'AG, l'OEP et API.

L'exploration de solutions est effectuée pour chacun des algorithmes par une recherche aléatoire guidée. Dans le cas de l'AG, cette exploration est effectuée par l'opérateur de mutation et, dans le cas de API, par l'opérateur d'exploration locale : ces deux opérateurs jouent des rôles très similaires. Dans l'OEP, l'exploration est réalisée grâce aux contributions aléatoires de ses composantes lors de la mise à jour du vecteur vitesse.

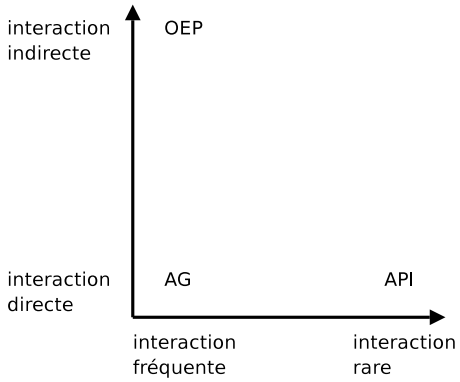


FIG. 4.4 – Positionnement de AG, OEP et API par rapport à la fréquence et au mode d'interaction.

Dans l'AG, l'exploitation des solutions est réalisée par renforcement statistique des zones d'intérêt, grâce à l'opérateur de sélection élitiste, chaque individu faisant partie de l'échantillonnage autour des zones d'intérêt. Dans l'OEP, l'exploitation des solutions est réalisée à travers la densité spatiale des particules autour des bonnes solutions. L'exploitation dans l'algorithme API est réalisée au moyen de deux mécanismes : la position du nid et le fouragement autour d'un site de chasse. Ces deux mécanismes peuvent être considérés comme un renforcement hiérarchique autour des zones d'intérêt. Le nid détermine le point focal de la zone d'intérêt autour de laquelle les fourmis établissent leurs sites de chasse. Le site de chasse d'une fourmi détermine sa zone d'exploration.

L'algorithme API possède une capacité spéciale que n'ont pas les deux autres algorithmes : il peut oublier des sites d'exploration non fructueux. Cet oubli est réalisé grâce à la patience sur les sites et le déplacement du nid. Cette capacité peut être un très grand avantage pour l'algorithme API, car elle lui permet d'abandonner une zone non profitable ou de se concentrer sur une autre zone, afin de ne pas gaspiller ses efforts.

4.5 Expérimentations

Après avoir exposé les différents algorithmes et avoir exploré les caractéristiques des métaheuristiques qui sont à leurs origines, nous nous attachons à l'étude expérimentale de ces algorithmes. Pour cela, nous procédons en deux étapes. Lors de la première étape, nous effectuons une recherche de « bons » paramètres, que nous utiliserons dans la deuxième partie de cette section pour évaluer les performances de ces algorithmes sur des cas plus complexes.

4.5.1 Détermination de « bons » paramètres pour les métaheuristiques

Avant de commencer, il est nécessaire de définir ce que nous entendons par « bons » paramètres. De « bons » paramètres sont des paramètres garantissant que la variable aléatoire associée à la probabilité du meilleur MMC trouvé possède une variance faible et une moyenne élevée. De « bons » paramètres sont également peu sensibles aux données d'entrée, telles que les séquences d'observations. Il convient de remarquer que les paramètres donnant une moyenne très élevée, mais une variance forte, ne seront pas forcément considérés comme de « bons » paramètres.

Afin de déterminer de « bons » paramètres pour chacun des algorithmes, nous avons réalisé de nombreux tests sur des valeurs possibles des paramètres sur un ensemble de quatre séquences d'observations issues d'images. Les performances de ces configurations de paramètres seront analysées à l'aide de graphiques particuliers que nous avons nommés PAPFP.

A l'aide de ces graphiques, nous serons en mesure de déduire de « bons » paramètres.

4.5.1.1 Les données d'expérimentation

Pour l'étude des algorithmes pour l'apprentissage de MMC, nous avons considéré plusieurs séquences d'observations issues de l'encodage d'images bitmap en niveaux de gris. Les images ayant servi à la construction des séquences d'observations sont celles employées dans (Samaria and Harter, 1994). De manière à limiter les temps de calcul déjà très importants en raison du nombre de tests réalisés, nous avons redimensionné les images au quart de leur taille. Ces petites images sont alors linéarisées en séquences. La linéarisation en une séquence d'une image consiste à la découper en blocs de 10 pixels de côté. Les lignes de pixels de chacun des blocs sont mise bout à bout, afin de former une séquence de pixels en niveaux de gris. Les séquences issues des blocs de l'image sont concaténées en considérant les blocs ligne par ligne du bloc le plus à gauche au bloc le plus à droite. La figure 4.5 synthétise la linéarisation d'une image. Ce type de codage a pour avantage de préserver une certaine proximité dans la séquence des pixels proches dans l'image. Ce mode de linéarisation d'une image n'est pas le seul utilisable : d'autres tels qu'un parcours de Peano (Mari et al., 2000) ou un codage en bande ont été utilisés dans la littérature. Nous avons choisi ce codage en raison de sa simplicité d'implémentation. Il n'est peut être pas optimal pour effectuer une reconnaissance de forme mais comme nous ne sommes intéressés que par l'apprentissage des MMC et non pas par l'étape de pré-traitement, ce choix n'a pour nous que peu d'influence. La base d'images ORL (Samaria and Harter, 1994) est composée de 10 photographies de visages de 40 personnes. Ne pouvant utiliser les 400 images pour nos tests, nous n'avons considéré que quatre images : la première photographie des quatre premières personnes de la base (cf. figure 4.6). Également pour réduire les temps nécessaires aux expérimentations, nous avons recodé les images en 32 niveaux de gris à l'aide d'une transformation linéaire. Le lecteur notera que ce ré-échelonnement des niveaux de gris est couramment effectué sur des images afin d'éviter que certains niveaux de gris n'apparaissant pas à l'apprentissage mais apparaissant à la reconnaissance, rendent le MMC trop intolérant. Ces quatre séquences seront nommées dans la suite Img_1, Img_2, Img_3 et Img_4.

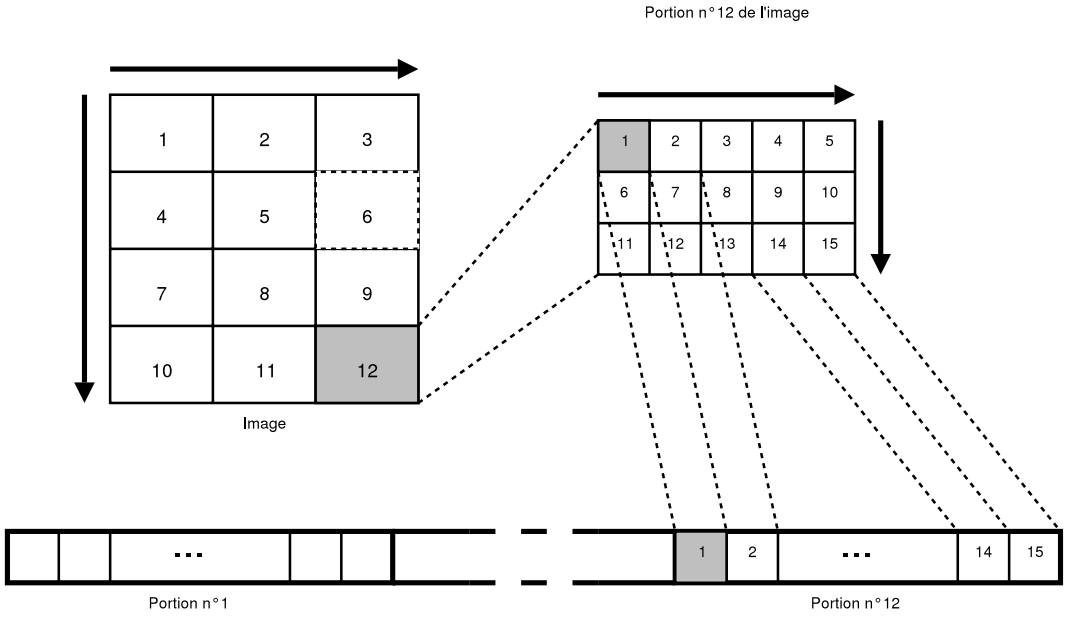


FIG. 4.5 – Principe de codage d'une image en séquences d'observation.

FIG. 4.6 – La première photographie des quatre premières personnes de la base ORL (Samaria and Harter, 1994).

4.5.1.2 Considérations générales et graphique PAPFP

Pour déterminer de bonnes configurations de paramètres pour chaque algorithme, nous procédons de la façon suivante. Les configurations de paramètres sont choisies de manière à explorer le même nombre de solutions. Cette approche permet de déterminer la façon dont l'algorithme doit se comporter pour explorer les solutions, notamment s'il faut beaucoup d'itérations ou beaucoup de particules/agents. 1000 solutions sont explorées lorsque le nombre d'itérations de Baum-Welch est non nul tandis que 30000 solutions sont explorées dans l'autre cas. Ces nombres de solutions laissent suffisamment de temps aux algorithmes pour converger presque totalement si ce n'est totalement. Lorsque l'algorithme de Baum-Welch n'est pas utilisé, il est nécessaire d'explorer beaucoup plus de solutions, car la recherche n'est pas guidée par une information de type gradient. Lorsque l'algorithme de Baum-Welch peut être utilisé, les expérimentations sont effectuées avec 0, 2 et 5 itérations de Baum-Welch. Les performances des configurations de paramètres sont étudiées séparément en fonction de ce nombre d'itérations. En effet, comme le nombre de solutions explorées est constant, les performances atteintes par un algorithme utilisant x ou y itérations de Baum-Welch sont forcément d'ordres de grandeur différents. 30 essais sont effectués par configuration de paramètres.

Nous proposons d'utiliser des graphes particuliers pour déterminer empiriquement quels sont les paramètres qui garantissent de bonnes performances. Nous avons nommé ces graphes « des graphes de Probabilité d'Apparitions de Paramètres en Fonction de la Performance ». Ces graphes sont abrégés dans la suite de ce chapitre par « graphes PAPFP ». Un graphe PAPFP permet, à partir d'un ensemble de configurations de paramètres X, d'estimer la probabilité d'apparition de la valeur v du paramètre p parmi les configurations X en fonction du logarithme de la probabilité obtenue. On note $p(x)$ la valeur du paramètre p et $f(x)$ le logarithme de la probabilité pour la configuration $x \in X$. Comme 30 essais sont effectués par configuration, chaque configuration de paramètres apparaît 30 fois dans X. Soit $\text{minf} = \min_{x \in X} f(x)$ et $\text{maxf} = \max_{x \in X} f(x)$. La fonction PAPFP pour le paramètre p et la valeur de paramètre v est

$$PAPFP(l)_{p(\cdot)=v} = \frac{1}{|X|} \sum_{x \in X, p(x)=v} \text{Normale}(f(x), 0.1 \cdot (\text{maxf} - \text{minf}))(l)$$

avec $\text{Normale}(m, \sigma)(l)$ la valeur prise par la loi normale de moyenne m et d'écart type σ en l. Une fonction PAPFP est proportionnelle à la somme de lois normales dont l'écart type est 10% de la plage de valeurs prise par la performance sur X. L'hypothèse effectuée dans la définition de ces courbes est que, pour une configuration $x \in X$, la valeur $f(x)$ associée est issue d'une loi normale centrée sur la valeur $f(x)$ d'écart-type $0.1 \cdot (\text{maxf} - \text{minf})$. Ainsi, si x est présent plusieurs fois[4] dans X, la moyenne de la fonction objectif sera estimée par la somme des lois normales. L'écart-type a été fixé à 10% de la plage de valeurs de $f(x)$ afin de réaliser un lissage des courbes et faire apparaître de manière plus nette les tendances suivies par les paramètres.

[4] C'est notre cas : 30 fois.

Le graphe PAPFP pour le paramètre p et l'ensemble de configurations X consiste alors à tracer les courbes $PAPFP(l)_{p(\cdot)=v)}$ en faisant varier la valeur v du paramètre. Dans les graphes suivants, nous noterons $PAPFP(l)$ la valeur prise par les courbes pour l fixé. Les graphes PAPFP possèdent plusieurs avantages :

- ils permettent d'obtenir un aperçu pertinent du profil de performances ;
- ils sont plus informatifs qu'une moyenne, un minimum, un maximum et/ou un écart type ;
- les profils de performances peuvent être plus complexes que des lois normales, sans nuire à l'interprétabilité des courbes.

Les résultats et graphes que nous présentons dans la suite sont similaires pour l'ensemble des séquences d'observations Img_1, Img_2, Img_3 et Img_4 sauf mentions contraires. Pour ne pas allonger inutilement ce chapitre, nous ne présenterons que les graphes obtenus pour Img_1.

Dans la suite de cette section, nous adoptons toujours la même démarche pour la détermination de ces « bons » paramètres. Ne pouvant évaluer l'interaction de tous les paramètres simultanément, nous procédons itérativement : du paramètre le plus significativement bon[5] à celui qui l'est le moins. Lorsqu'un paramètre est choisi, les graphes PAPFP sont tracés à nouveau en ne considérant que les éléments de X possédant ce ou ces paramètres. Si aucun paramètre n'est significativement meilleur ou si plusieurs paramètres semblent significatifs simultanément, plusieurs choix sont explorés en parallèle, aboutissant à plusieurs configurations de paramètres.

4.5.1.3 La métaheuristique AG

Pour la recherche des bons paramètres de AG, nous avons considéré les valeurs de paramètres définies dans la table 4.3.

TAB. 4.3 – Paramètres de l'algorithme AG.

Paramètres	Domaines de valeurs
$\mathcal{N}$	4, 5, 10, 20, 50, 100, 200
$\mathcal{T}_{\max}$	$1000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} > 0$ $30000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} = 0$
$\mathcal{N}_{\text{BW}}$	0, 2, 5
$\mathcal{N}_{\text{parent}}$	$\mathcal{N}/2$
p_{mut}	0.0, 0.01, 0.1, 0.3, 0.5
MuterParents	Oui, Non
OptimiserParents	Oui, Non si $\mathcal{N}_{\text{BW}} > 0$ Non si $\mathcal{N}_{\text{BW}} = 0$

Les graphes PAPFP obtenus pour $\mathcal{N}_{\text{BW}} = 0$ sont donnés par la figure 4.7. Ces graphiques nous informent que les meilleurs paramètres correspondent à *MuterParents*=Non et $p_{\text{mut}} = 0.01$. Le graphe PAPFP du paramètre $\mathcal{N}$ sous ces conditions est donné par la figure 4.8. Il apparaît que $\mathcal{N} = 5$ est le meilleur choix et qu'augmenter le nombre d'individus dans la population diminue les performances. De même, lorsque ce nombre diminue.

Les graphes PAPFP obtenus pour $\mathcal{N}_{\text{BW}} = 2$ sont donnés par la figure 4.9. En utilisant une démarche similaire à celle suivie précédemment, nous trouvons que deux bonnes configurations de paramètres apparaissent :

[5]Nous considérons qu'un paramètre est significativement meilleur qu'un autre si sa moyenne est plus élevée et son écart-type est plus réduit.

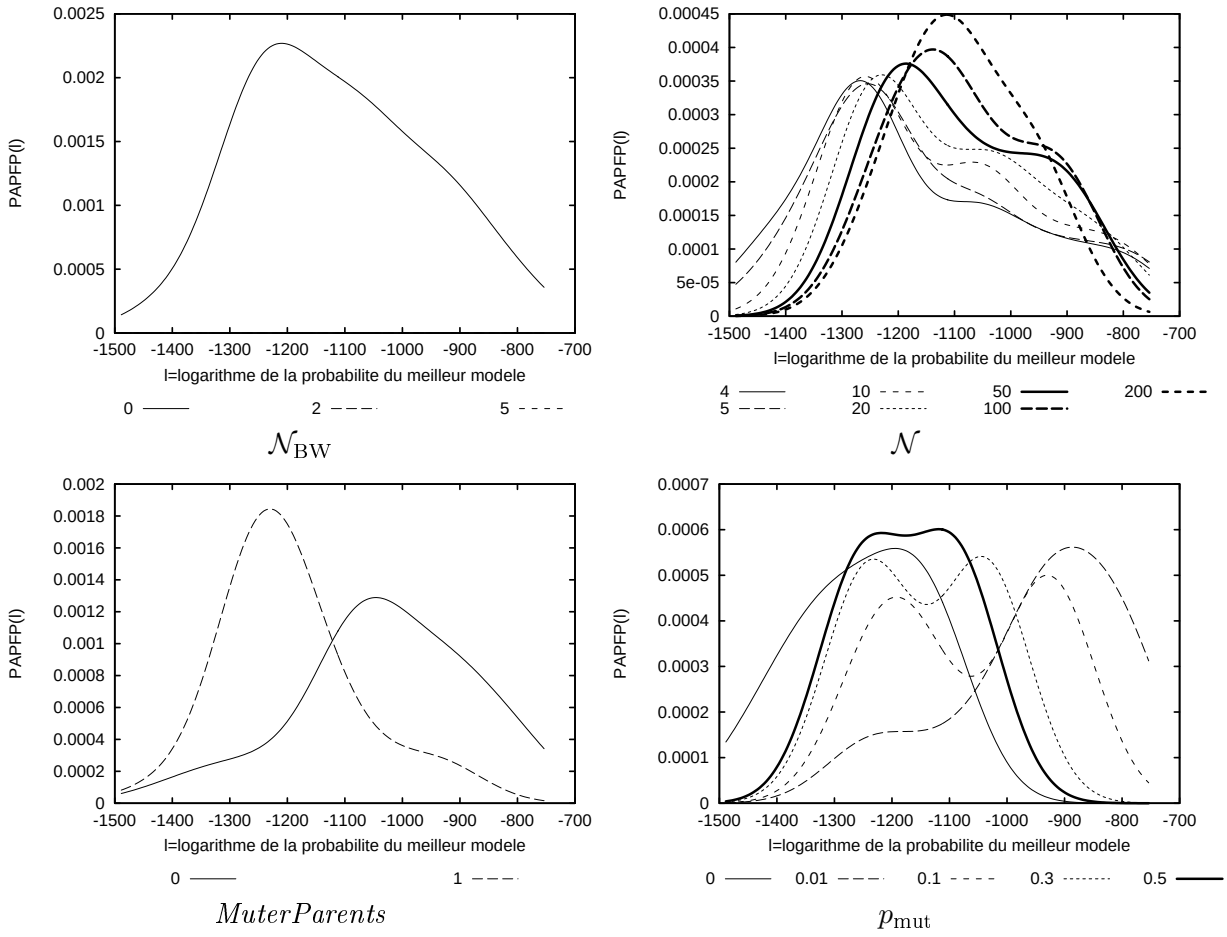


FIG. 4.7 – PAPFP de AG lorsque $\mathcal{N}_{\mathrm{BW}} = 0$.

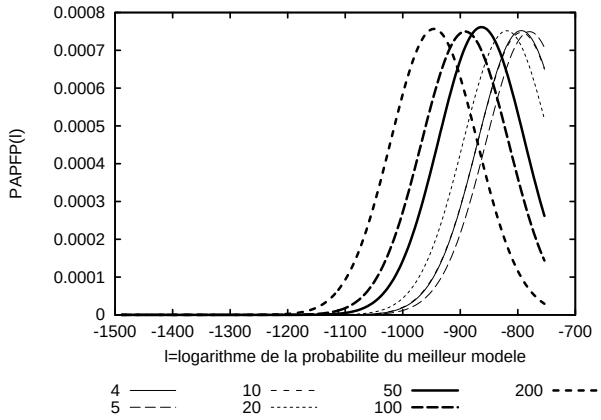


FIG. 4.8 – PAPFP de AG pour $\mathcal{N}$ lorsque $\mathcal{N}_{\mathrm{BW}} = 0$, *MuterParents*=Non et $p_{\mathrm{mut}} = 0.01$.

– $\mathcal{N} = 20$, $p_{\mathrm{mut}} = 0.3$, *MuterParents*=Non et *OptimiserParents*=Oui
– $\mathcal{N} = 5$, $p_{\mathrm{mut}} = 0.3$, *MuterParents*=Oui et *OptimiserParents*=Oui

Les graphes PAPFP obtenus pour $\mathcal{N}_{\mathrm{BW}} = 5$ sont donnés par la figure 4.10. En utilisant une démarche similaire à celle suivie jusqu'ici, nous trouvons que deux configurations de paramètres se démarquent :

– $\mathcal{N} = 20$, $p_{\mathrm{mut}} = 0.5$, *MuterParents*=Oui et *OptimiserParents*=Oui
– $\mathcal{N} = 5$, $p_{\mathrm{mut}} = 0.5$, *MuterParents*=Non et *OptimiserParents*=Oui

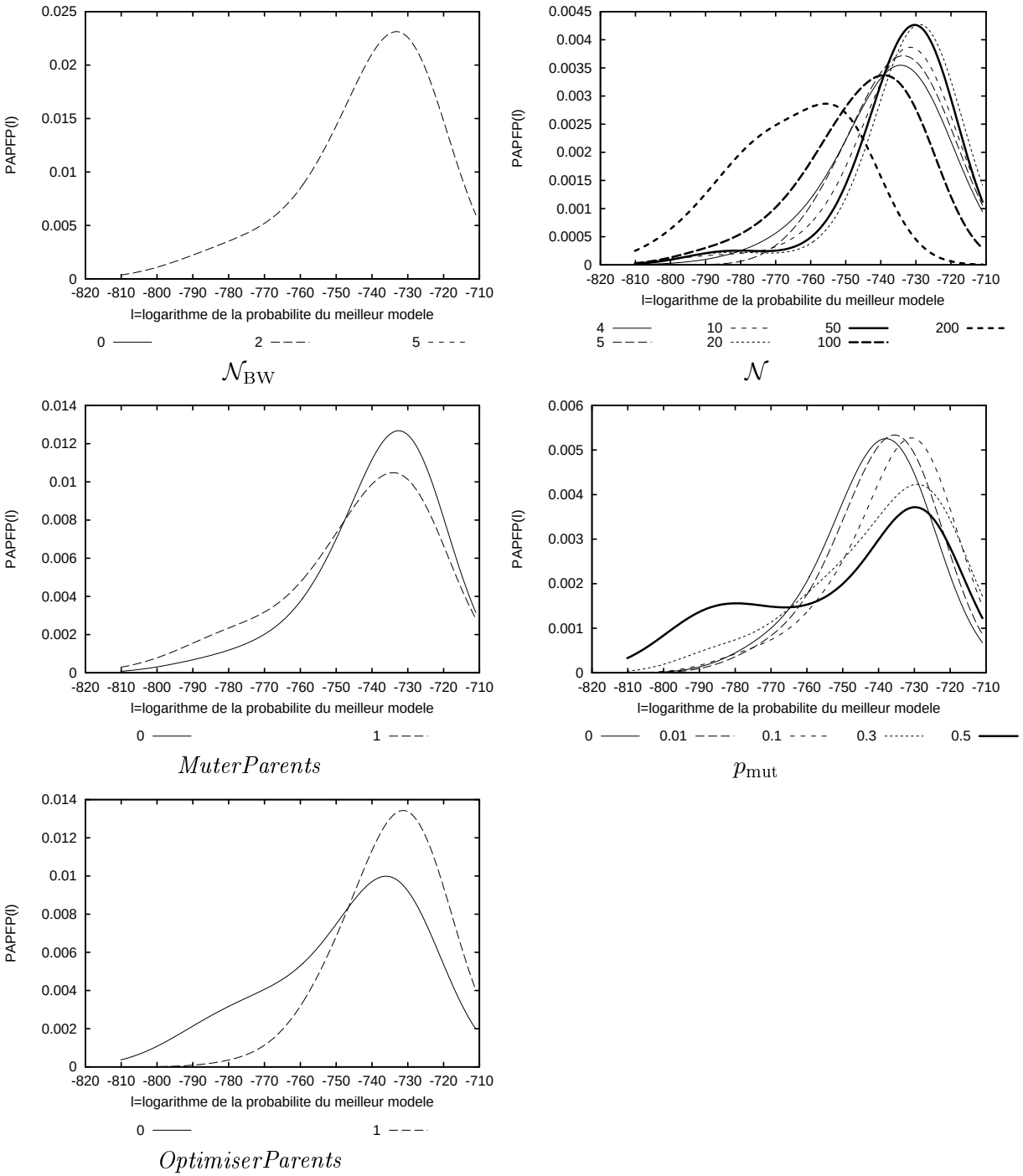


FIG. 4.9 – PAPFP de AG lorsque $\mathcal{N}_{\mathrm{BW}} = 2$.

Nous pouvons donc considérer les cinq configurations de paramètres suivantes comme de « bonnes » configurations de paramètres :

- AG1 : *MuterParents*=Non, $p_{\mathrm{mut}} = 0.01$, $\mathcal{N} = 5$
- AG2 : $\mathcal{N} = 5$, $p_{\mathrm{mut}} = 0.3$, *MuterParents*=Oui et *OptimiserParents*=Oui
- AG3 : $\mathcal{N} = 20$, $p_{\mathrm{mut}} = 0.3$, *MuterParents*=Non et *OptimiserParents*=Oui
- AG4 : $\mathcal{N} = 20$, $p_{\mathrm{mut}} = 0.5$, *MuterParents*=Oui et *OptimiserParents*=Oui
- AG5 : $\mathcal{N} = 5$, $p_{\mathrm{mut}} = 0.5$, *MuterParents*=Non et *OptimiserParents*=Oui

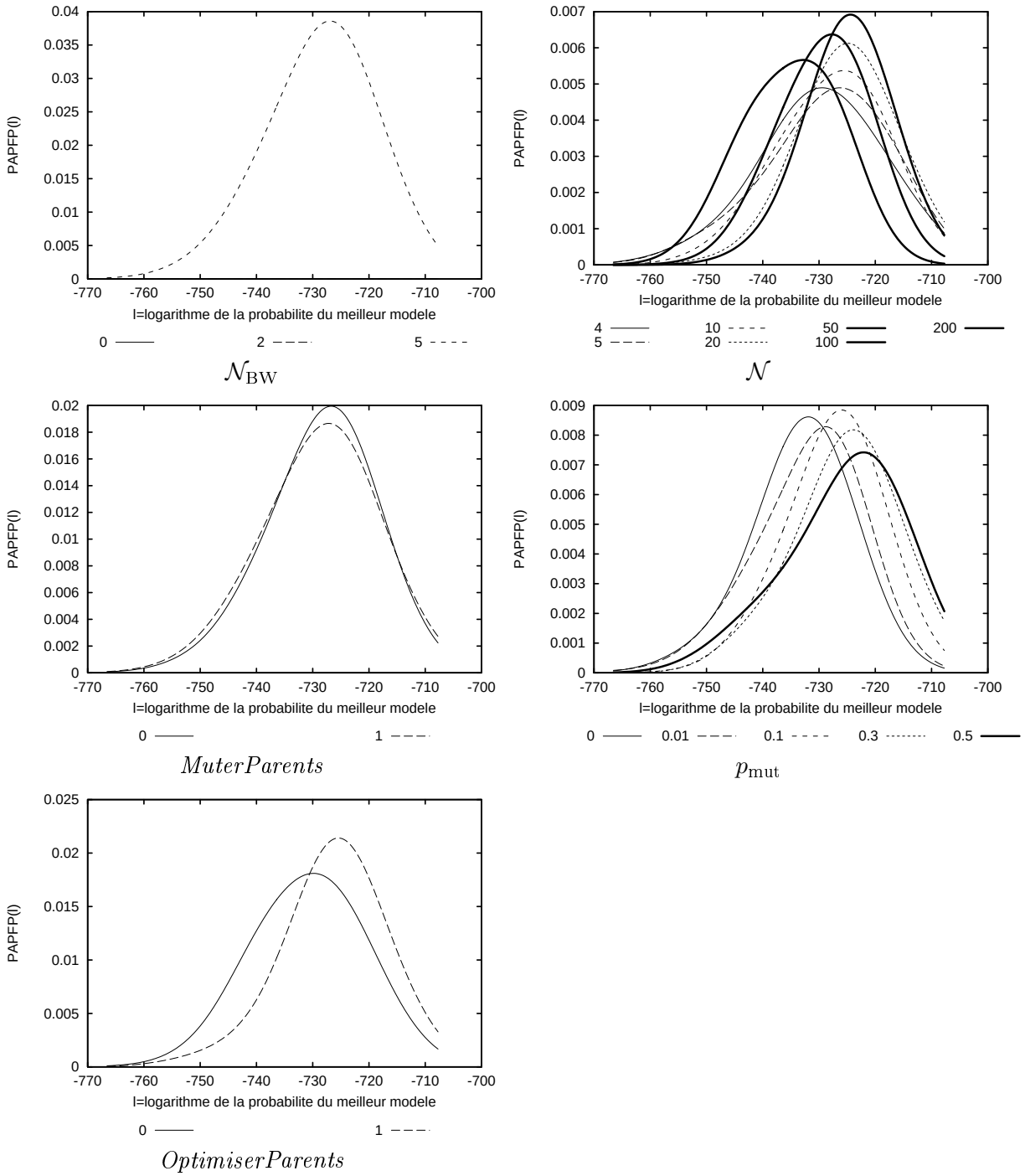


FIG. 4.10 – PAPFP de AG lorsque $\mathcal{N}_{\mathrm{BW}} = 5$.

On remarque qu'aucune règle pour la définition des paramètres génériques ne semble émerger si ce n'est concernant la probabilité de mutation : plus le nombre d'itérations de Baum-Welch est important, plus il est nécessaire d'augmenter la probabilité de mutation p_{mut} afin d'augmenter la diversité des individus de la population.

4.5.1.4 La métaheuristique API

Étude du paramètre $M_{\max}$

Nous commençons cette étude par le paramètre $M_{\max}$, pour lequel il est possible de montrer qu'augmenter le nombre de fourmis ou augmenter la taille de la mémoire est statistiquement équivalent.

Pour cela, nous avons effectué l'apprentissage des séquences d'observations Art_1 [6] et Img_1 200 fois à l'aide de l'algorithme API avec un paramétrage hétérogène (cf. section 2.7.2 et plus loin). Les paramètres utilisés sont :

- nombre d'itérations de Baum-Welch : $\mathcal{N}_{\text{BW}} \in \{0, 2, 5\}$
- taille de la mémoire des fourmis : $M_{\max} \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$
- le nombre de fourmis : NbFourmis $= 100/M_{\max}$
- le nombre d'itérations de l'algorithme : $T_{\max} = 2000/\text{NbFourmis}$ pour Art_1 et $T_{\max} = 5000/\text{NbFourmis}$ pour Img_1
- la patience locale : $e_{\max} = 3$
- la patience du nid : $T_{\text{Déplacement}} = Entier(1.5 * e_{\max} * M_{\max})$

Ces paramètres possèdent plusieurs propriétés :

- le nombre de solutions explorées pour $\mathcal{N}_{\text{BW}}$ fixé est constant.
- le nombre de sites de chasse explorés simultanément pour $\mathcal{N}_{\text{BW}}$ fixé est constant et égal à $100 = M_{\max} * \text{NbFourmis}$
- le nombre de déplacements du nid pour $\mathcal{N}_{\text{BW}}$ fixé est constant. Sa valeur est fixée à $\frac{T_{\max}}{T_{\text{Déplacement}}}$ c'est-à-dire 90 pour Art_1 et 225 pour Img_1.

Pour chaque valeur prise par $\mathcal{N}_{\text{BW}}$, nous avons effectué le test non paramétrique de la moyenne sur chaque paire de paramètres. Les résultats du test sur les 12 configurations de paramètres sont donnés par la figure 4.11. Le test effectué a été réalisé avec un seuil de 0.01.

Cette figure montre que, dans la grande majorité des cas, augmenter le nombre de fourmis ou augmenter la taille mémoire est statistiquement équivalent. Quelques rares tests ont échoué, probablement en raison des arrondis dûs au fait que les paramètres sont des entiers. Afin de faciliter l'étude des paramètres de l'algorithme API, nous fixons la taille de la mémoire à un (1) de manière à éviter les interactions entre les autres paramètres et la taille de la mémoire.

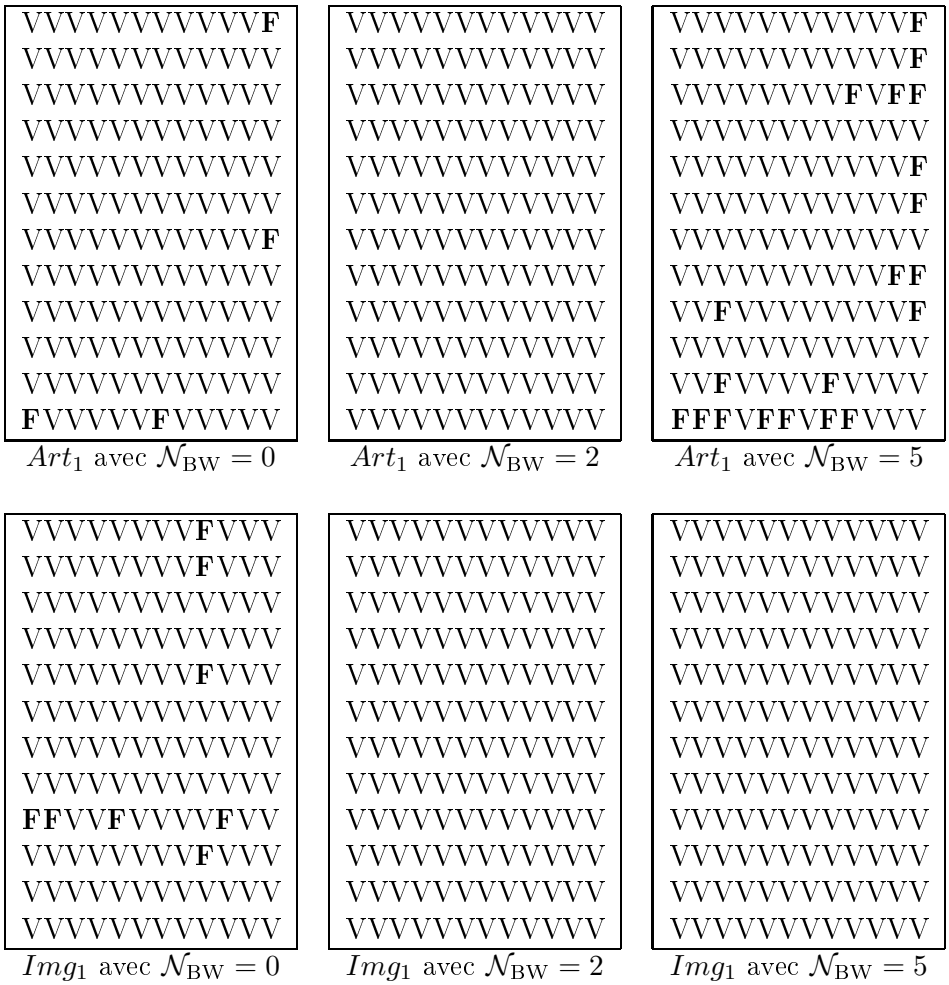
Etude des paramètres

Pour la recherche des bons paramètres de API pour l'apprentissage de MMC, nous avons considéré deux types de paramétrages : un paramétrage homogène, pour lequel toutes les fourmis possèdent les mêmes caractéristiques et un paramétrage hétérogène, lorsque les caractéristiques des fourmis sont différentes. Nous noterons APIHomo la première forme de paramétrage et APIHete la deuxième forme. Les paramètres expérimentés pour l'algorithme API sont donnés par la table 4.4.

Etude des configurations de paramètres APIHomo

La figure 4.12 présente les graphes PAPFP lorsque $\mathcal{N}_{\text{BW}} = 0$ obtenu pour Img_1. On remarque que les meilleures performances sont obtenues pour $\mathcal{A}^i_{\text{local}} = \mathcal{A}^i_{\text{site}} = 0.1$ et qu'une augmentation de ces valeurs réduit de manière significative les performances. On remarque sous ces conditions que $e_{\max} = 1$ et de manière assez peu significative que $\mathcal{T}_{\text{Déplacement}} = 15$ et $\mathcal{N} = 50$.

[6] L'observation Art_1 est reprise de (Monmarché, 2000). Elle possède $M =$ symboles et est définie par $Art_1 = (s_1, s_2, s_3, s_1, s_4, s_2, s_4, s_4)$.



Les abscisses (gauche vers droite) et les ordonnées (haut vers bas) correspondent à la taille de la mémoire des fourmis *i.e.* $M_{\max} \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$.
V représente le cas où les performances ne sont pas significativement différentes tandis que F (en gras) représente le cas où le test a échoué et les performances peuvent être considérées significativement différentes.

FIG. 4.11 – Résultat du test de la médiane au seuil de 0.01.

La figure 4.13 présente les graphes PAPFP obtenus lorsque $\mathcal{N}_{\mathrm{BW}} = 2$ pour Img_1. On remarque que les meilleures performances sont obtenues pour $\mathcal{A}^i_{\mathrm{local}} = \mathcal{A}^i_{\mathrm{site}} = 0.1$ et qu'une augmentation de ces valeurs réduit de manière significative les performances. On remarque sous ces conditions que $\mathcal{N} = 5$ et qu'augmenter ou diminuer ce nombre diminue les performances. Finalement, on trouve de manière assez peu significative que $\mathcal{T}_{\mathrm{Déplacement}} = 15$ et $e_{\max} = 4$.

La figure 4.14 présente les graphes PAPFP obtenus lorsque $\mathcal{N}_{\mathrm{BW}} = 5$ pour Img_1. On remarque que les meilleures performances sont encore une fois obtenues pour $\mathcal{A}^i_{\mathrm{local}} = \mathcal{A}^i_{\mathrm{site}} = 0.1$ et qu'une augmentation de ces valeurs réduit de manière significative les performances. On remarque sous ces conditions que $\mathcal{N} = 20$ et qu'augmenter ou diminuer ce nombre diminue les performances. Finalement, on trouve de manière assez peu significative que $\mathcal{T}_{\mathrm{Déplacement}} = 15$ et $e_{\max} = 4$.

TAB. 4.4 – Paramètres de l'algorithme API.

APIHomo	
Paramètres	Domaines de valeurs
$\mathcal{N}$	2, 5, 10, 20, 50, 100
$\mathcal{T}_{\max}$	$1000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} > 0$ $30000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} = 0$
$\mathcal{N}_{\text{BW}}$	0, 2, 5
$\mathcal{M}_{\max}$	1
$e_{\max}$	1, 2, 3, 4, 5
$\mathcal{T}_{\text{Déplacement}}$	5, 10, 15, 20
$\mathcal{A}^i_{\text{site}}$	0.1, 0.2, 0.3, 0.4, 0.5
$\mathcal{A}^i_{\text{local}}$	0.1, 0.2, 0.3, 0.4, 0.5

APIHete	
Paramètres	Domaines de valeurs
$\mathcal{N}$	2, 5, 10, 20, 50, 100
$\mathcal{T}_{\max}$	$1000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} > 0$ $30000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} = 0$
$\mathcal{N}_{\text{BW}}$	0, 2, 5
$\mathcal{M}_{\max}$	1
$e_{\max}$	1, 2, 3, 4, 5
$\mathcal{T}_{\text{Déplacement}}$	5, 10, 15, 20
$\mathcal{A}^i_{\text{site}}$	$0.01(\frac{1}{0.01}^{\frac{i}{\mathcal{N}}})$
$\mathcal{A}^i_{\text{local}}$	$0.01(\frac{1}{0.01}^{\frac{i}{\mathcal{N}}})/10$

Nous pouvons donc considérer les trois configurations de paramètres suivantes comme étant optimales :

- APIHomo1 : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 50$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.1$, $e_{\max} = 1$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHomo2 : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 5$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.1$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHomo3 : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 20$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.1$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$

La propriété suivante se dégage de ces expérimentations : l'amplitude d'exploration locale et de site doivent être petites.

Etude des configurations de paramètres APIHete

La figure 4.15 présente les graphes PAPFP obtenus lorsque $\mathcal{N}_{\text{BW}} = 0$ pour Img_1. Nous remarquons que le paramètre le plus discriminant est $\mathcal{N}$, le nombre de fourmis. La valeur permettant d'obtenir la meilleure performance est $\mathcal{N} = 2$. Lorsque l'on trace les graphes PAPFP avec $\mathcal{N} = 2$, le paramètre $e_{\max}$ (cf. figure 4.16) devient le plus significatif et indique que sa valeur doit être $e_{\max} = 5$. Finalement, sous ces conditions ($\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 2$ et $e_{\max} = 5$), on remarque que $\mathcal{T}_{\text{Déplacement}} = 10$.

La figure 4.17 présente les graphes PAPFP obtenus lorsque $\mathcal{N}_{\text{BW}} = 2$ pour Img_1. De même que précédemment, nous remarquons que le paramètre le plus significatif est $\mathcal{N}$ qui doit pendre pour valeur 5. On remarque également que plus $\mathcal{N}$ est grand, moins les performances sont bonnes, mais un nombre trop petit de fourmis est également néfaste. Les graphes PAPFP obtenus lorsque $\mathcal{N}_{\text{BW}} = 2$ et $\mathcal{N} = 5$ mettent en évidence que $\mathcal{T}_{\text{Déplacement}} = 15$ et $e_{\max} = 4$.

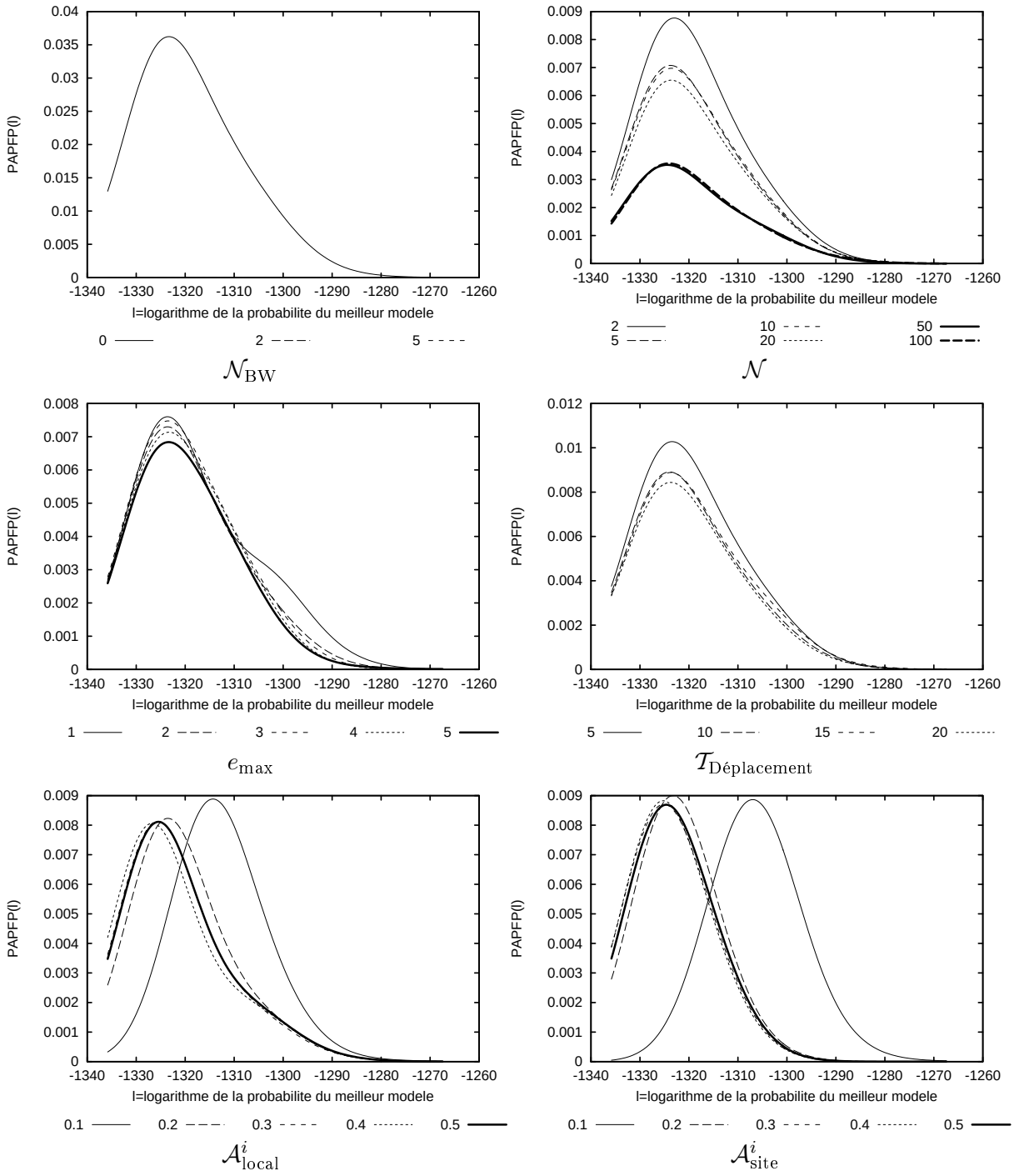


FIG. 4.12 – PAPFP de APIHomo lorsque $\mathcal{N}_{\mathrm{BW}} = 0$.

La figure 4.18 présente les graphes PAPFP obtenus lorsque $\mathcal{N}_{\mathrm{BW}} = 5$ pour Img_1. De même que précédemment, nous remarquons que le paramètre le plus significatif est $\mathcal{N}$ qui doit prendre pour valeur 5. On remarque que le comportement des courbes est très similaire au cas $\mathcal{N}_{\mathrm{BW}} = 2$. Les graphes PAPFP de $e_{\max}$ et $\mathcal{T}_{\mathrm{Déplacement}}$ lorsque $\mathcal{N}_{\mathrm{BW}} = 5$ et $\mathcal{N} = 5$ sont donnés par la figure 4.19. Ces graphes montrent que $e_{\max} = 3$ et $\mathcal{T}_{\mathrm{Déplacement}} = 5$.

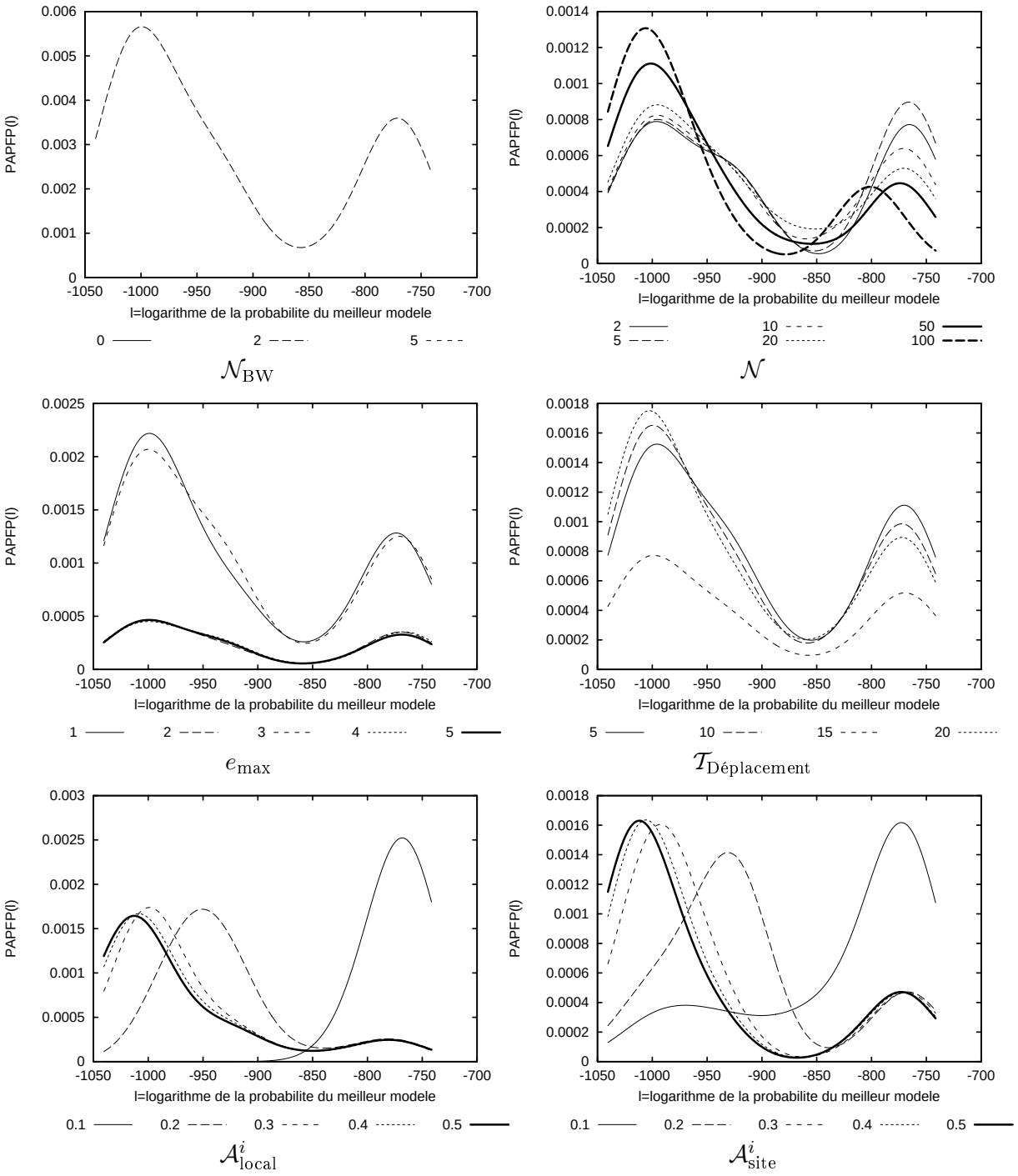


FIG. 4.13 – PAPFP de APIHomo lorsque $\mathcal{N}_{\mathrm{BW}} = 2$.

Nous pouvons donc considérer les trois configurations de paramètres suivantes comme étant optimales :

- APIHete1 : $\mathcal{N}_{\mathrm{BW}} = 0$, $\mathcal{N} = 2$, $e_{\max} = 5$, $\mathcal{T}_{\mathrm{Déplacement}} = 10$
- APIHete2 : $\mathcal{N}_{\mathrm{BW}} = 2$, $\mathcal{N} = 5$, $e_{\max} = 4$, $\mathcal{T}_{\mathrm{Déplacement}} = 15$
- APIHete3 : $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 5$, $e_{\max} = 3$, $\mathcal{T}_{\mathrm{Déplacement}} = 5$

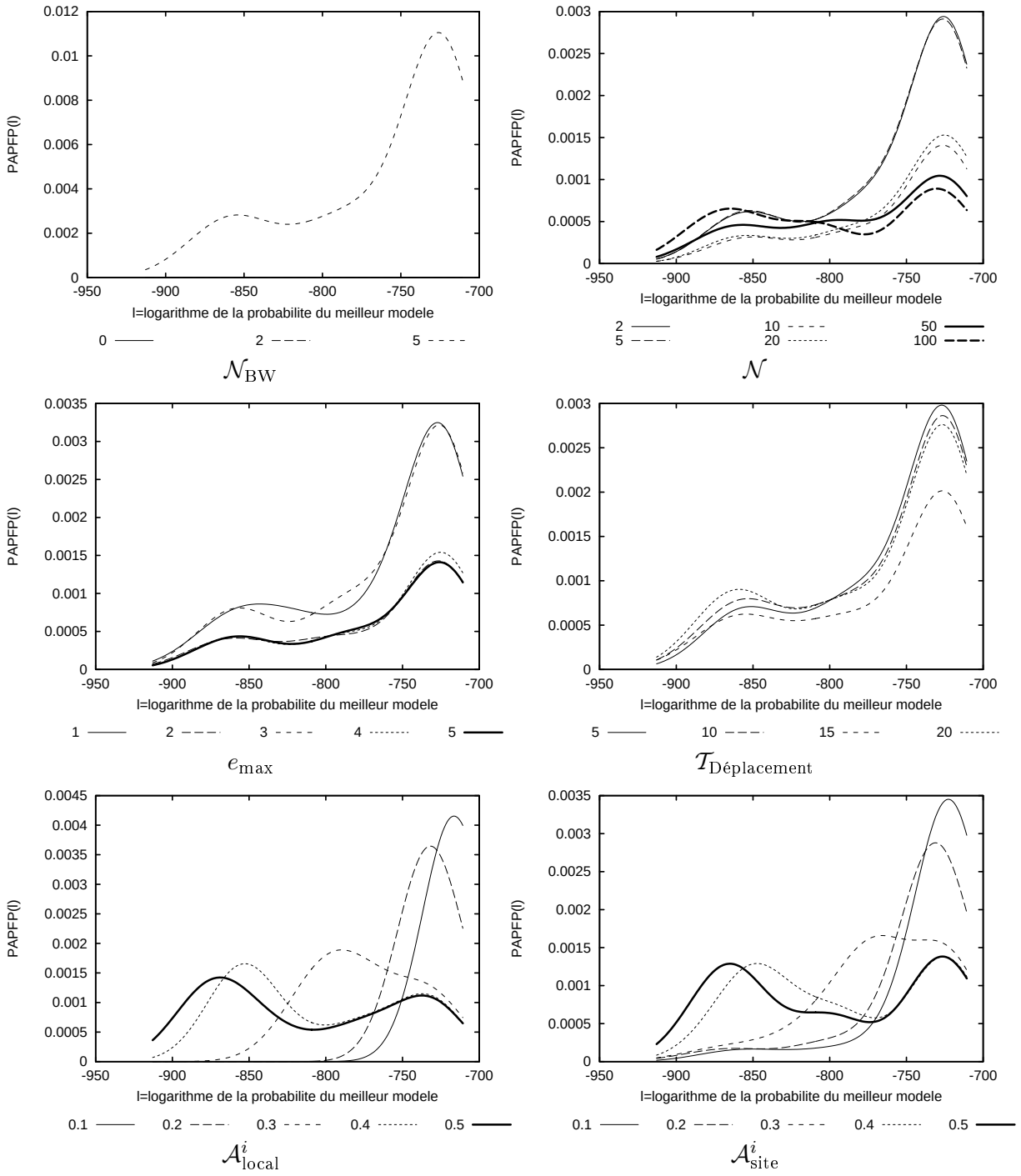


FIG. 4.14 – PAPFP de APIHomo lorsque $\mathcal{N}_{\mathrm{BW}} = 5$.

Plusieurs propriétés émergent de ces expérimentations lorsque les meilleures performances sont obtenues :

- il vaut mieux augmenter le nombre d'itérations de l'algorithme qu'augmenter le nombre de fourmis ;
- plus on utilise d'itérations de Baum-Welch, moins il est nécessaire d'être patient sur un site de chasse.

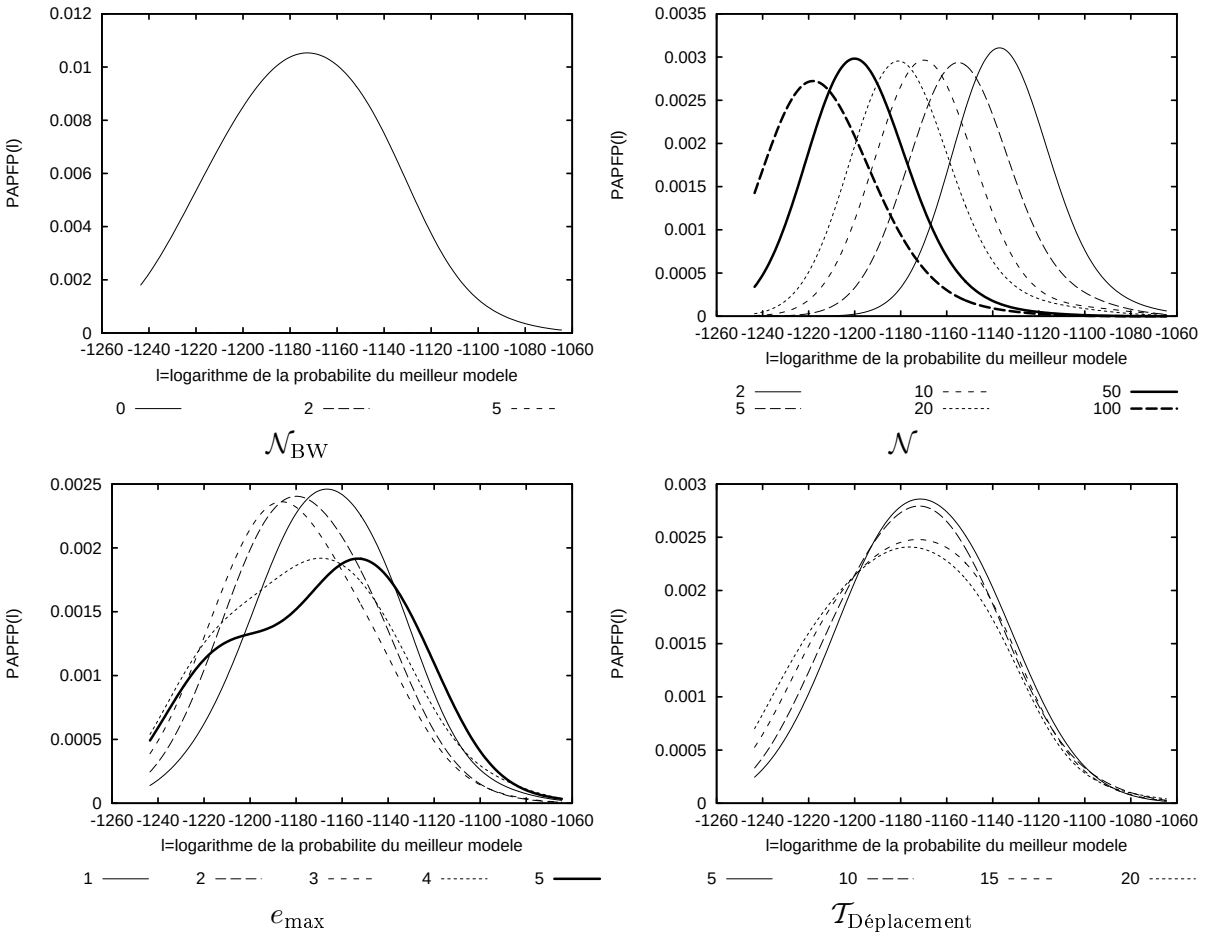


FIG. 4.15 – PAPFP de APIHete lorsque $\mathcal{N}_{\mathrm{BW}} = 0$.

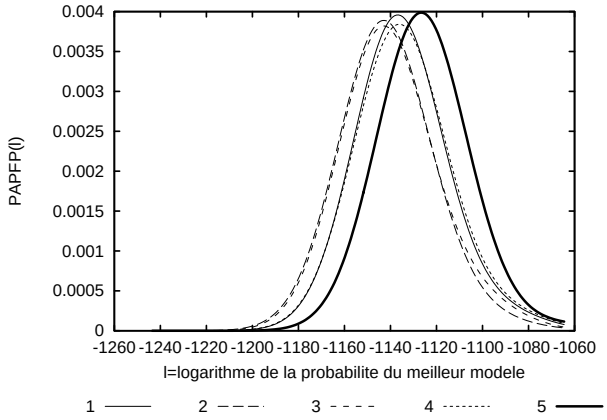


FIG. 4.16 – PAPFP de APIHete pour $e_{\max}$ lorsque $\mathcal{N}_{\mathrm{BW}} = 0$ et $\mathcal{N} = 2$.

4.5.1.5 La métaheuristique OEPDistance

Pour la recherche des bons paramètres, nous avons considéré les configurations de paramètres définies dans la table 4.5.

Dans cette adaptation de l'OEP, nous utilisons un voisinage des particules dépendant d'une mesure de distance d. Soient $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ deux MMC de Ω. Soient $\lambda^{(1)} = (A^{(1)}, B^{(1)}, \Pi^{(1)})$ et $\lambda^{(2)} = (A^{(2)}, B^{(2)}, \Pi^{(2)})$ les MMC associés dans Λ^*.

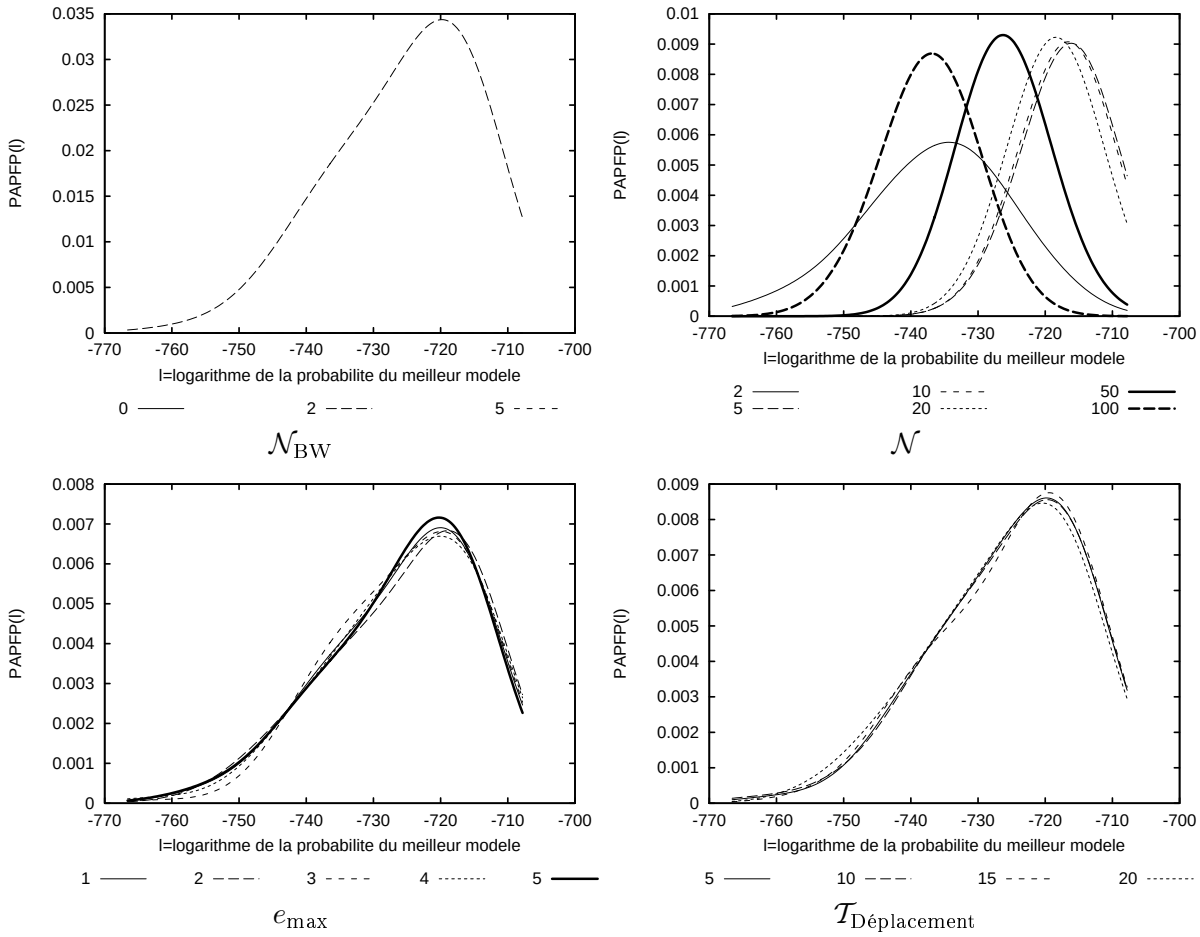


FIG. 4.17 – PAPFP de APIHete lorsque $\mathcal{N}_{\mathrm{BW}} = 2$.

TAB. 4.5 – Paramètres de l'algorithme OEPDistance.

Paramètres	Domaines de valeurs
$\mathcal{N}$	4, 5, 10, 20, 50, 100, 200
$\mathcal{T}_{\max}$	$1000/\mathcal{N}$ si $\mathcal{N}_{\mathrm{BW}} > 0$ $30000/\mathcal{N}$ si $\mathcal{N}_{\mathrm{BW}} = 0$
ω	0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6
c_1	0, 0.5, 1, 1.5, 2
c_2	0, 0.5, 1, 1.5, 2
d	d_0, d_1, d_2, d_3
V	3 si $d \neq d_0$ $\mathcal{N}$ sinon

Dans la littérature, on peut trouver de nombreuses distances pour des MMC. La distance la plus simple que nous avons considérée et nommée d_1 est une distance matricielle pondérée par le nombre de variables du MMC sur l'espace Λ (Falkhausen et al., 1995).

$$d_1(\lambda^{(1)}, \lambda^{(2)}) = \sqrt{\frac{||A^{(1)} - A^{(2)}||^2}{N \cdot M} + \frac{||B^{(1)} - B^{(2)}||^2}{N^2} + \frac{||\Pi^{(1)} - \Pi^{(2)}||^2}{N}}$$

en désignant par $||X||^2$ le carré de la distance euclidienne sur les coefficients des matrices. La

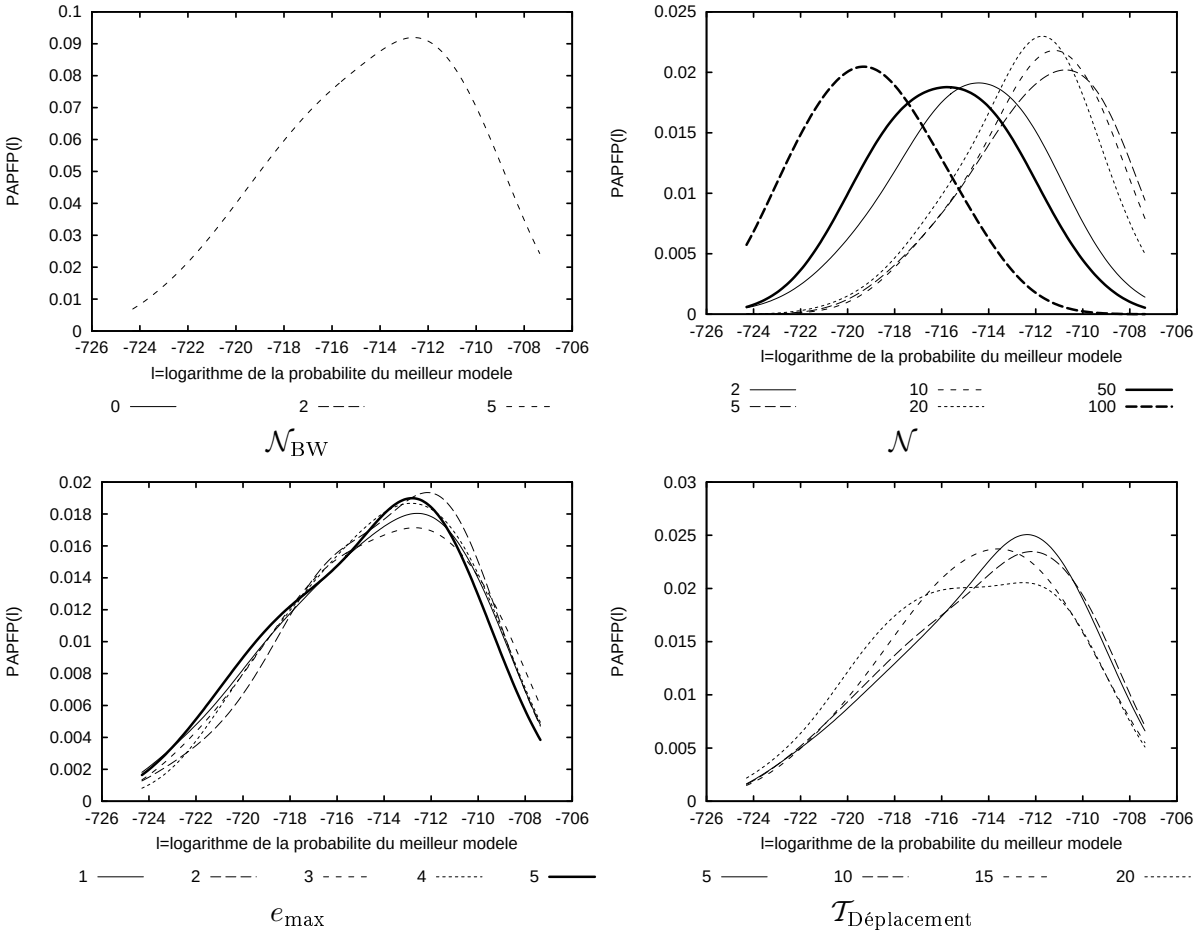


FIG. 4.18 – PAPFP de APIHete lorsque $\mathcal{N}_{\mathrm{BW}} = 5$.

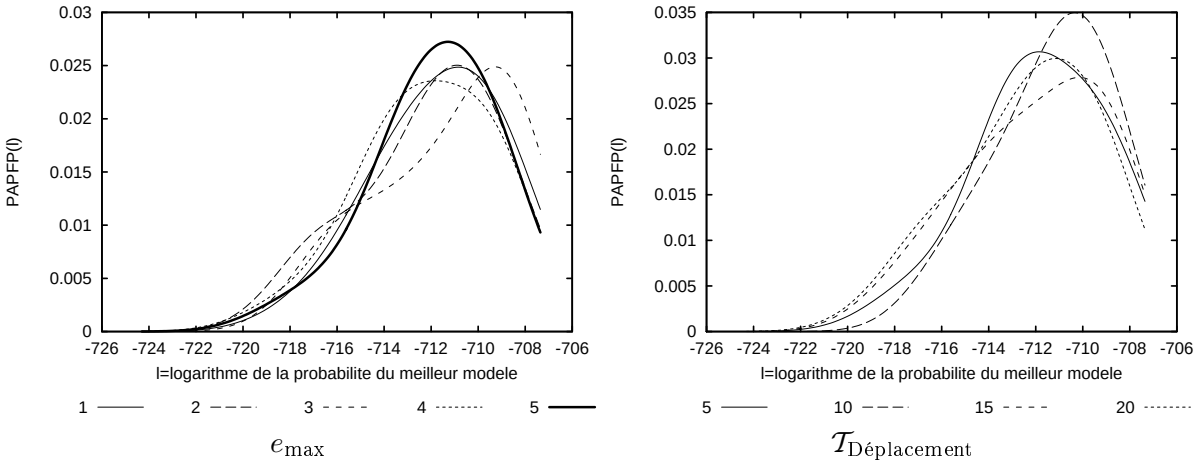


FIG. 4.19 – PAPFP de APIHete lorsque $\mathcal{N}_{\mathrm{BW}} = 5$ et $\mathcal{N} = 5$.

complexité de ce calcul est $O(N^2 + NM)$.

La deuxième distance que nous avons considérée est basée sur la notion d'entropie relative des distributions de probabilités. La définition de cette distance également connue sous le nom de Kullback-Leibler (KL) (Rabiner, 1989) est :

$$d_{\mathrm{KL}}(\lambda^{(1)}, \lambda^{(2)}) = \int_O \frac{1}{G(O)} P(O|\lambda^{(1)}) \ln \frac{P(O|\lambda^{(2)})}{P(O|\lambda^{(1)})} \partial O$$

en désignant par $G(O)$ une fonction proportionnelle à la longueur T de la séquence O.

Le calcul de cette distance est très coûteux en temps car il nécessite l'utilisation de la technique de Monte-Carlo (Falkhausen et al., 1995) pour estimer sa valeur. Or cette distance doit être calculée de nombreuses fois entre plusieurs particules, rendant difficile son emploi pour l'apprentissage de MMC. Une solution consiste à utiliser une approximation. Plusieurs approximations existent dans la littérature (Do, 2003) (Vihola et al., 2002). Nous avons considéré celle décrite dans (Do, 2003) définie par :

$$d_{\mathrm{FKL}}(\lambda^{(1)}, \lambda^{(2)}) = \sum_{i=1}^{N} \nu_i \left(D(\mathbf{a}_i^{(1)}, \mathbf{a}_i^{(2)}) + D(\mathbf{b}_i^{(1)}, \mathbf{b}_i^{(2)}) \right)$$

avec $\nu = (\nu_i)'_{1 \leq i \leq N}$ l'unique distribution de probabilité vérifiant $\nu' A^{(1)} = \nu'$[7] et

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} x_i \log \frac{x_i}{y_i} \text{ with } \mathbf{x} = (x_i)_{1 \leq i \leq N} \text{ et } \mathbf{y} = (y_i)_{1 \leq i \leq N}$$

Cette mesure n'étant pas symétrique, nous avons considéré :

$$d_2(\lambda^{(1)}, \lambda^{(2)}) = \frac{d_{\mathrm{FKL}}(\lambda^{(1)}, \lambda^{(2)}) + d_{\mathrm{FKL}}(\lambda^{(2)}, \lambda^{(1)})}{2}$$

La complexité[8] du calcul de cette distance est $O(N^4 + NM)$.

La dernière distance que nous avons considérée est définie sur Ω par :

$$d_3(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = ||\mathbf{x}^{(1)} \ominus \mathbf{x}^{(2)}||_2$$

avec $||\mathbf{x}||_2$ la norme euclidienne usuelle. La complexité de ce calcul est $O(N^2 + NM)$.

Du point de vue de l'efficacité des calculs, nous pouvons remarquer que le calcul du voisinage peut utiliser la distance ou son carré sans changer le comportement de l'algorithme. Le calcul de la racine carrée n'est donc pas effectué. De plus, il est nécessaire en fonction de l'espace de travail Λ^* ou Ω et de la distance d'effectuer la conversion de modèles d'un espace vers l'autre. Cette conversion a un coût de $O(N^2 + NM)$.

Ne pouvant tester de trop nombreuses configurations de paramètres, nous avons limité la taille du voisinage à 3 pour les distances d_1, d_2 et d_3. Nous avons également considéré une version gbest de cet algorithme avec $V = \mathcal{N}$. Ce type de paramétrage ne dépendant pas de la distance utilisée, nous utilisons la distance d_0 prenant toujours la valeur 0.

Les graphes PAPFP obtenus lorsque $\mathcal{N}_{\mathrm{BW}} = 0$ sont donnés par la figure 4.20. Ces graphes nous informent que les bons paramètres sont $\mathcal{N} = 100$, $\omega = 1$, $c_1 = 0.5$, $c_2 = 0.5$, $d = d_0$. On remarquera que les différences entre les distances ne sont pas significatives et que n'importe laquelle peut donc être utilisée.

Les graphes PAPFP obtenus lorsque $\mathcal{N}_{\mathrm{BW}} = 2$ sont donnés par la figure 4.24. Ces graphes nous informent que les bons paramètres sont $\mathcal{N} = 20$, $\omega = 0.4$, $c_1 = 1$, $c_2 = 0$. On remarque que $c_2 = 0$ permet d'obtenir les meilleures performances, par conséquent le voisinage des particules ne semble pas utile pour la recherche de bonnes solutions.

Les graphes PAPFP obtenus lorsque $\mathcal{N}_{\mathrm{BW}} = 5$ sont donnés par la figure 4.22. Ces graphes nous informent que les bons paramètres sont $\mathcal{N} = 50$, $\omega = 0.2$, $c_1 = 0$, $c_2 = 0$. On remarque que $c_2 = 0$ permet d'obtenir les meilleures performances, par conséquent le voisinage des particules ne semble pas utile pour la recherche de bonnes solutions. On remarque également que les particules n'ont pas besoin de la mémoire de la meilleure position qu'elles ont vue.

Nous pouvons donc considérer les trois configurations de paramètres suivantes comme étant de « bonnes » configurations de paramètres :

[7] Pour un modèle de Markov stationnaire, on a $\lim_{n \to \infty} \pi' A^n = \nu'$.

[8] Le terme N^4 est dû à l'estimation de ν.

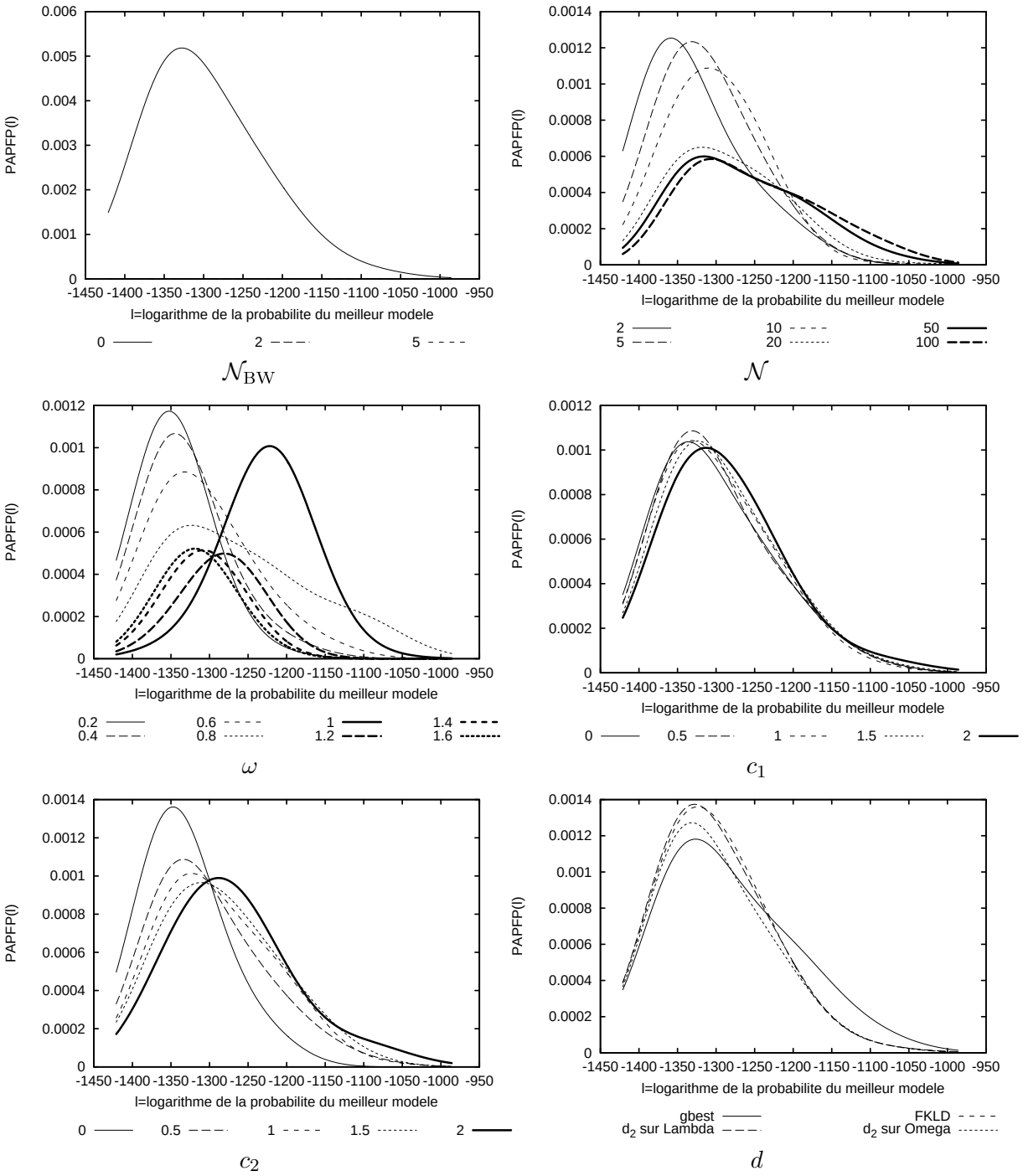


FIG. 4.20 – PAPFP de OEPDistance lorsque $\mathcal{N}_{\mathrm{BW}} = 0$.

- $\mathcal{N}_{\mathrm{BW}} = 0$, $\mathcal{N} = 100$, $\omega = 1$, $c_1 = 0.5$, $c_2 = 0.5$, $d = d_0$
- $\mathcal{N}_{\mathrm{BW}} = 2$, $\mathcal{N} = 20$, $\omega = 0.4$, $c_1 = 1$, $c_2 = 0$
- $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 50$, $\omega = 0.2$, $c_1 = 0$, $c_2 = 0$, $V = 6$

On remarque alors que, lorsque l'algorithme de Baum-Welch est utilisé, l'utilisation d'un voisinage réduit significativement les performances de l'algorithme. On remarque également que ω diminue lorsque le nombre d'itérations de Baum-Welch augmente.

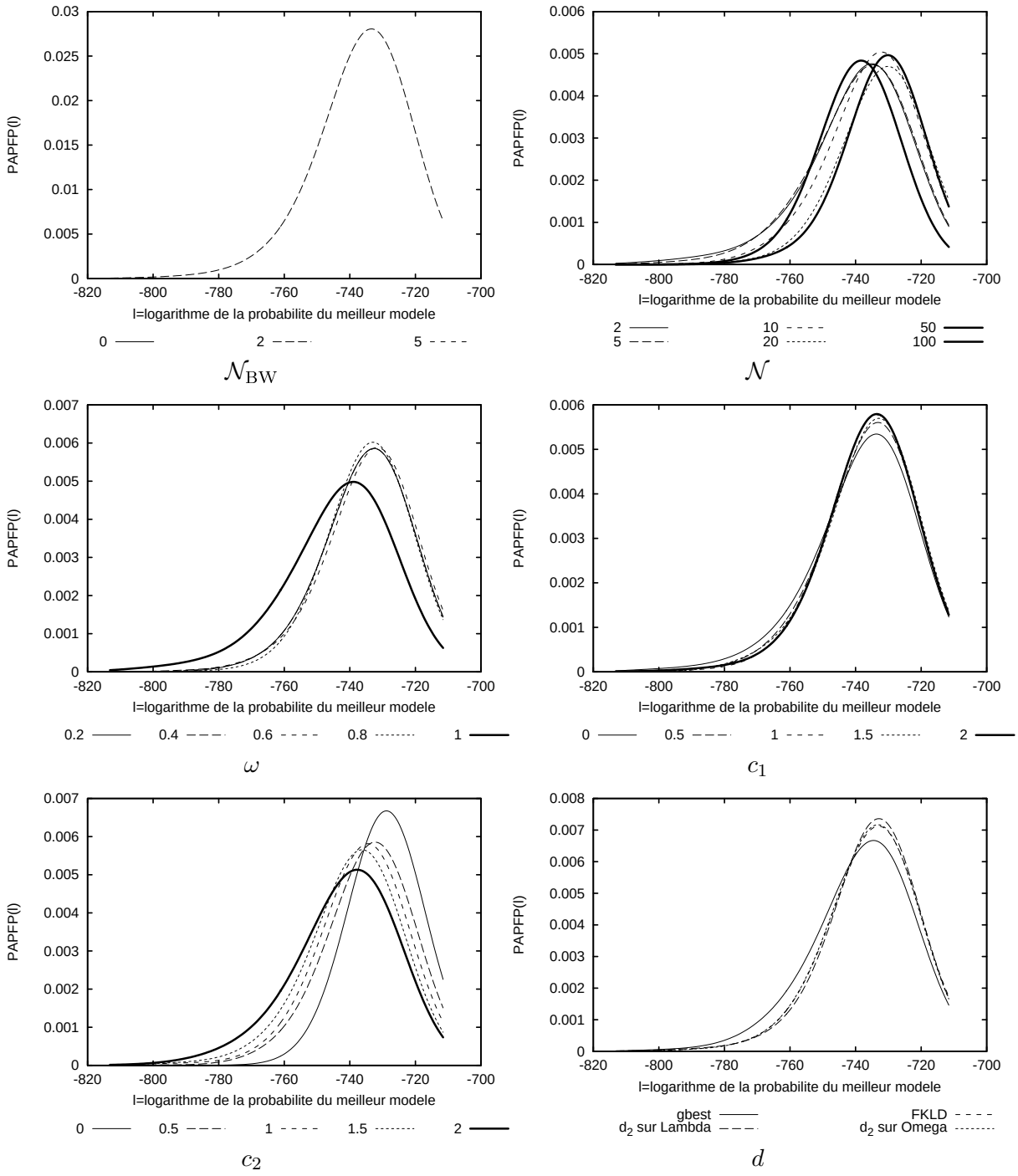


FIG. 4.21 – PAPFP de OEPDistance lorsque $\mathcal{N}_{\mathrm{BW}} = 2$.

4.5.1.6 La métaheuristique OEPSocial

Pour la recherche des bons paramètres, nous avons considéré les configurations de paramètres définies dans la table 4.6.

Les graphes PAPFP obtenus lorsque $\mathcal{N}_{\mathrm{BW}} = 0$ sont donnés par la figure 4.23. Ces graphes nous informent que les bons paramètres sont $\mathcal{N} = 20$, $\omega = 0.8$, $c_1 = 1.5$, $c_2 = 2.5$, $V = 9$. On remarquera un changement de comportement des PAPFP pour le paramètre ω à la valeur 1 montrant qu'il serait probablement profitable de borner l'amplitude des vecteurs vitesses,

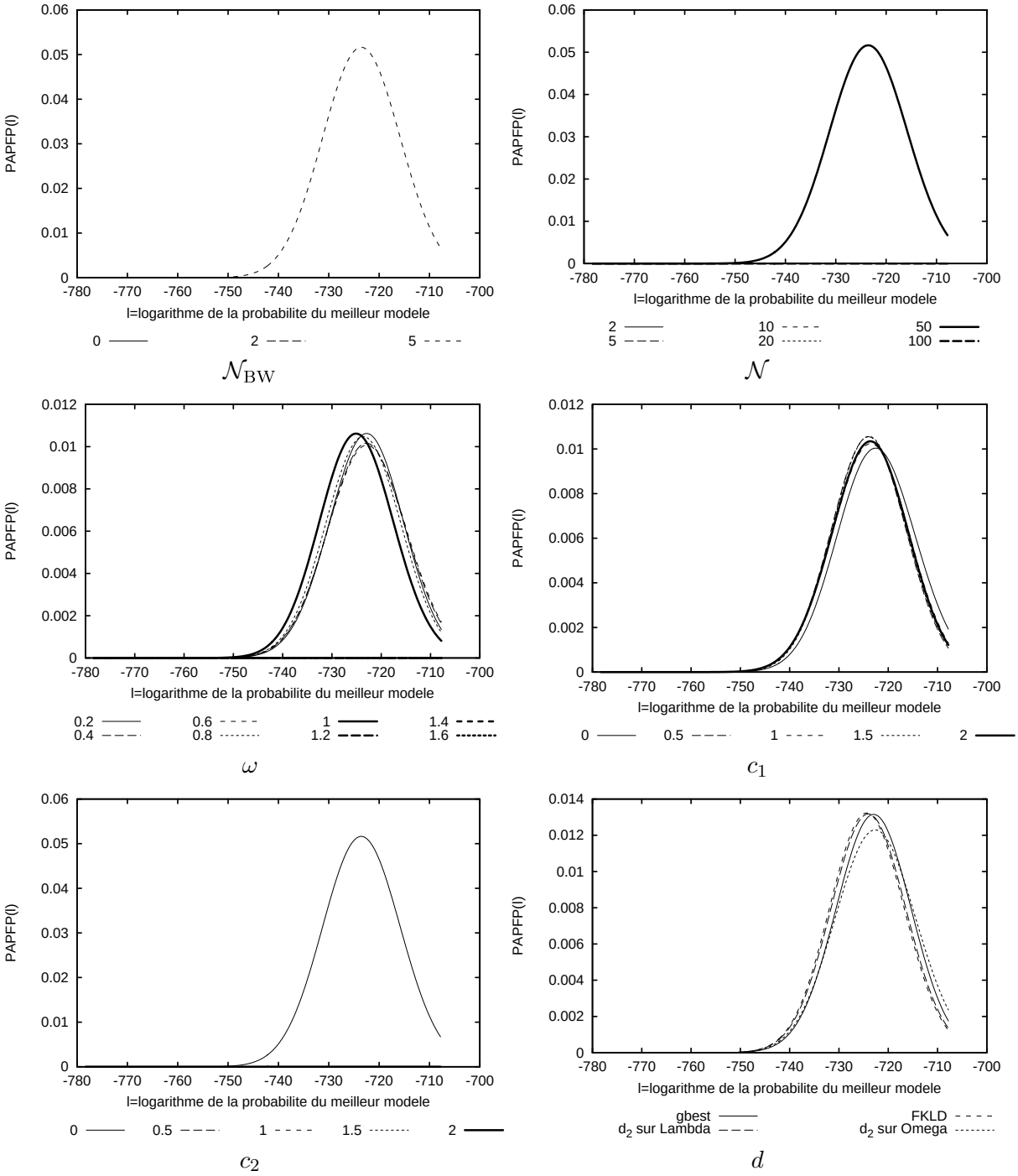


FIG. 4.22 – PAPFP de OEPDistance lorsque $\mathcal{N}_{\mathrm{BW}} = 5$.

afin d'éviter une probable divergence[9].

Les graphes PAPFP obtenus lorsque $\mathcal{N}_{\mathrm{BW}} = 2$ sont donnés par la figure 4.24. Ces graphes nous informent que les bons paramètres sont $\mathcal{N} = 20$, $\omega = 0.8$, $c_1 = 1.5$, $c_2 = 0$. On remarque que $c_2 = 0$ permet d'obtenir les meilleures performances, par conséquent le voisinage des particules ne semble pas utile pour la recherche de bonnes solutions.

[9]Pour tout $(p, x) \in (\Omega_K)^2$, on peut montrer que la limite $\lim_{||x||_2 \to \infty} \phi_K(p + x) = g(p)$ est un vecteur stochastique de $\mathbb{G}_K$ ne comportant que des zéros et un seul un (1). La divergence amène le système vers les extrêmes de l'espace Λ^*, desquels l'algorithme de Baum-Welch ne peut s'échapper.

TAB. 4.6 – Paramètres de l'algorithme OEPSocial.

Paramètres	Domaines de valeurs
$\mathcal{N}$	4, 5, 10, 20, 50, 100, 200
$\mathcal{T}_{\max}$	$1000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} > 0$ $30000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} = 0$
ω	0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6
c_1	0, 0.5, 1, 1.5, 2
c_2	0, 0.5, 1, 1.5, 2
V	3, 6, 9

Les graphes PAPFP obtenus lorsque $\mathcal{N}_{\text{BW}} = 5$ sont donnés par la figure 4.25. Ces graphes nous informent que les bons paramètres sont $\mathcal{N} = 50$, $\omega = 0.4$, $c_1 = 0$, $c_2 = 0.5$, $V = 6$. On remarque que $c_2 = 0$ permet d'obtenir les meilleures performances, par conséquent le voisinage des particules ne semble pas utile pour la recherche de bonnes solutions.

Nous pouvons donc considérer les trois configurations de paramètres suivantes comme étant de « bonnes » configurations de paramètres :

- OEPSocial1 : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 20$, $\omega = 0.8$, $c_1 = 1.5$, $c_2 = 2.5$, $V = 9$
- OEPSocial2 : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 20$, $\omega = 0.8$, $c_1 = 1.5$, $c_2 = 0$
- OEPSocial3 : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 50$, $\omega = 0.4$, $c_1 = 0$, $c_2 = 0.5$, $V = 6$

On peut remarquer un changement de comportement dans le fonctionnement de l'algorithme en fonction du nombre d'itérations de Baum-Welch. Lorsque peu d'itérations ($\mathcal{N}_{\text{BW}} = 2$) de Baum-Welch sont utilisées, seule l'utilisation du voisinage semble fructueuse, alors que, dans le cas $\mathcal{N}_{\text{BW}} = 5$, seule l'exploitation de la mémoire de la particule l'est.

4.5.1.7 La métaheuristique AGDiscret

Pour la recherche des bons paramètres, nous avons considéré les configurations de paramètres définies dans la table 4.7.

TAB. 4.7 – Paramètres de l'algorithme AGDiscret.

Paramètres	Domaines de valeurs
$\mathcal{N}$	4, 5, 10, 20, 50, 100, 200
$\mathcal{T}_{\max}$	$1000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} > 0$ $30000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} = 0$
$\mathcal{N}_{\text{parent}}$	$\mathcal{N}/2$
p_{mut}	0.0, 0.01, 0.1, 0.3, 0.5
MuterParents	Oui, Non

Ces graphiques nous informent que les meilleurs paramètres correspondent à *MuterParents*=Non et $p_{\text{mut}} = 0.01$. Le graphe PAPFP du paramètre $\mathcal{N}$ sous ces conditions est donné par la figure 4.27. Il apparaît que $\mathcal{N} = 5$ est le meilleur choix et qu'augmenter le nombre d'individus dans la population diminue les performances. De même, lorsque ce nombre diminue.

Nous pouvons donc considérer la configuration de paramètres suivante comme étant optimale : AGDiscret : *MuterParents*=Non, $p_{\text{mut}} = 0.01$, $\mathcal{N} = 5$.

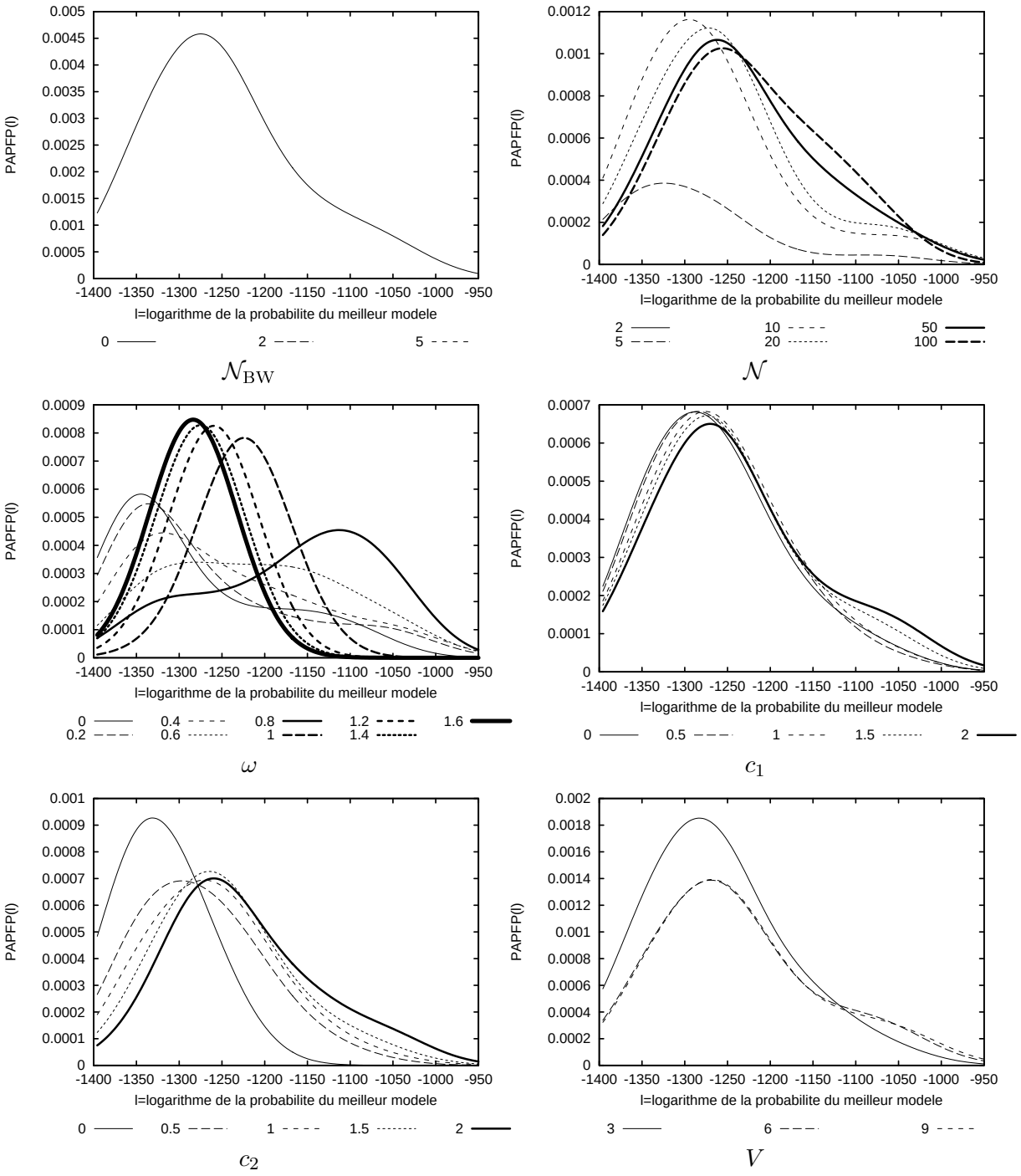


FIG. 4.23 – PAPFP de OEPSocial lorsque $\mathcal{N}_{\mathrm{BW}} = 0$.

4.5.1.8 La métaheuristique APIDiscret

Pour la recherche des bons paramètres de APIDiscret pour l'apprentissage de MMC, nous avons considéré deux types de paramétrages : un paramétrage homogène, pour lequel toutes les fourmis possèdent les mêmes caractéristiques et un paramétrage hétérogène lorsque les caractéristiques des fourmis sont différentes. Nous noterons APIDiscretHomo la première forme de paramétrage et APIDiscretHete la deuxième forme. Les paramètres expérimentés pour l'algorithme API sont donnés par la table 4.8.

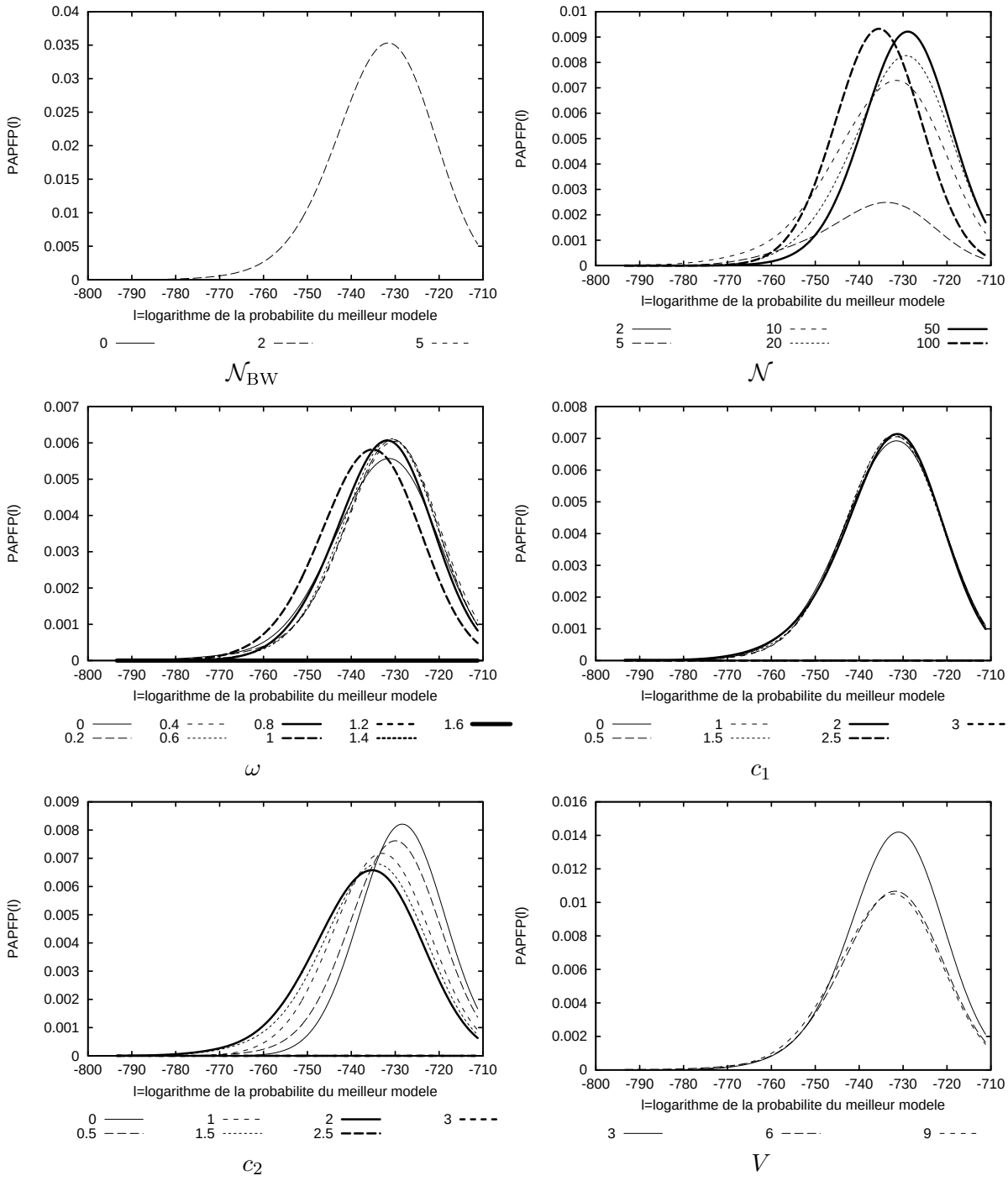


FIG. 4.24 – PAPFP de OEPSocial lorsque $\mathcal{N}_{\mathrm{BW}} = 2$.

Étude des configurations de paramètres APIDiscretHomo

La figure 4.28 présente les graphes PAPFP obtenus pour Img_1. Nous remarquons que le paramètre le plus discriminant est $\mathcal{A}^i_{\mathrm{site}} = 0.1$. Sous ces conditions, et de manière itérative, du paramètre le plus significatif au moins significatif, on trouve $e_{\max} = 1, \mathcal{N} = 2, \mathcal{T}_{\mathrm{Déplacement}} = 5$ et $\mathcal{A}^i_{\mathrm{local}} = 0.2$

Nous pouvons donc considérer la configuration de paramètres suivante comme étant optimale : APIDiscretHomo : $\mathcal{A}^i_{\mathrm{site}} = 0.1$, $e_{\max} = 1$, $\mathcal{N} = 2$, $\mathcal{T}_{\mathrm{Déplacement}} = 5$ et $\mathcal{A}^i_{\mathrm{local}} = 0.2$.

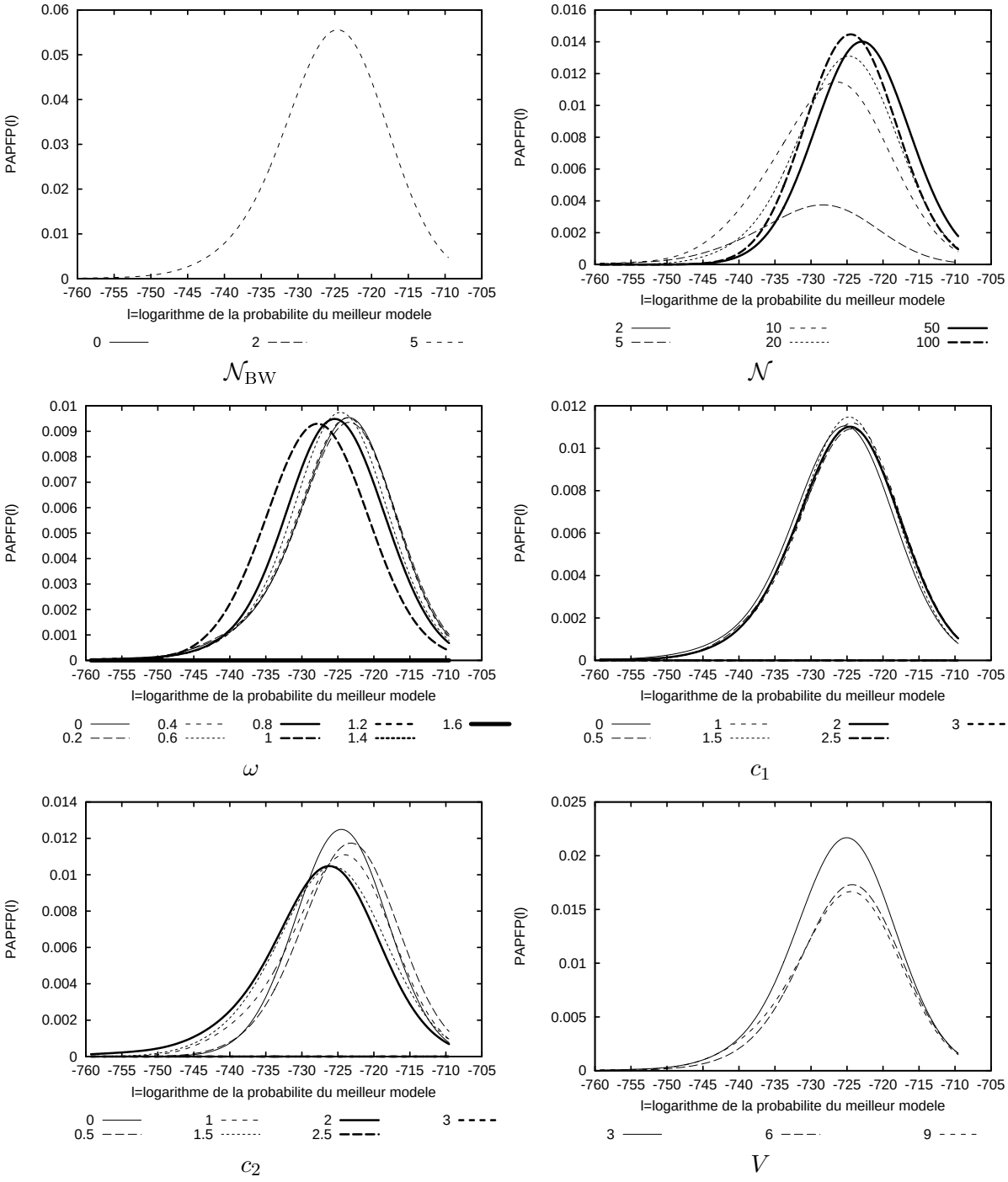


FIG. 4.25 – PAPFP de OEPSocial lorsque $\mathcal{N}_{\mathrm{BW}} = 5$.

Etude des configurations de paramètres APIDiscretHete

La figure 4.29 présente les graphes PAPFP obtenus pour Img_1. Nous remarquons que le paramètre le plus discriminant est $\mathcal{N}$, le nombre de fourmis. La valeur permettant d'obtenir la meilleure performance est $\mathcal{N} = 2$. Lorsque l'on trace les graphes PAPFP (cf. figure 4.30) avec $\mathcal{N} = 2$, il vient que $e_{\max} = 1$ et $\mathcal{T}_{\text{Déplacement}} = 5$.

Nous pouvons donc considérer la configuration de paramètres suivante comme étant optimale : APIDiscretHete : $e_{\max} = 1$, $\mathcal{N} = 2$, $\mathcal{T}_{\text{Déplacement}} = 5$.

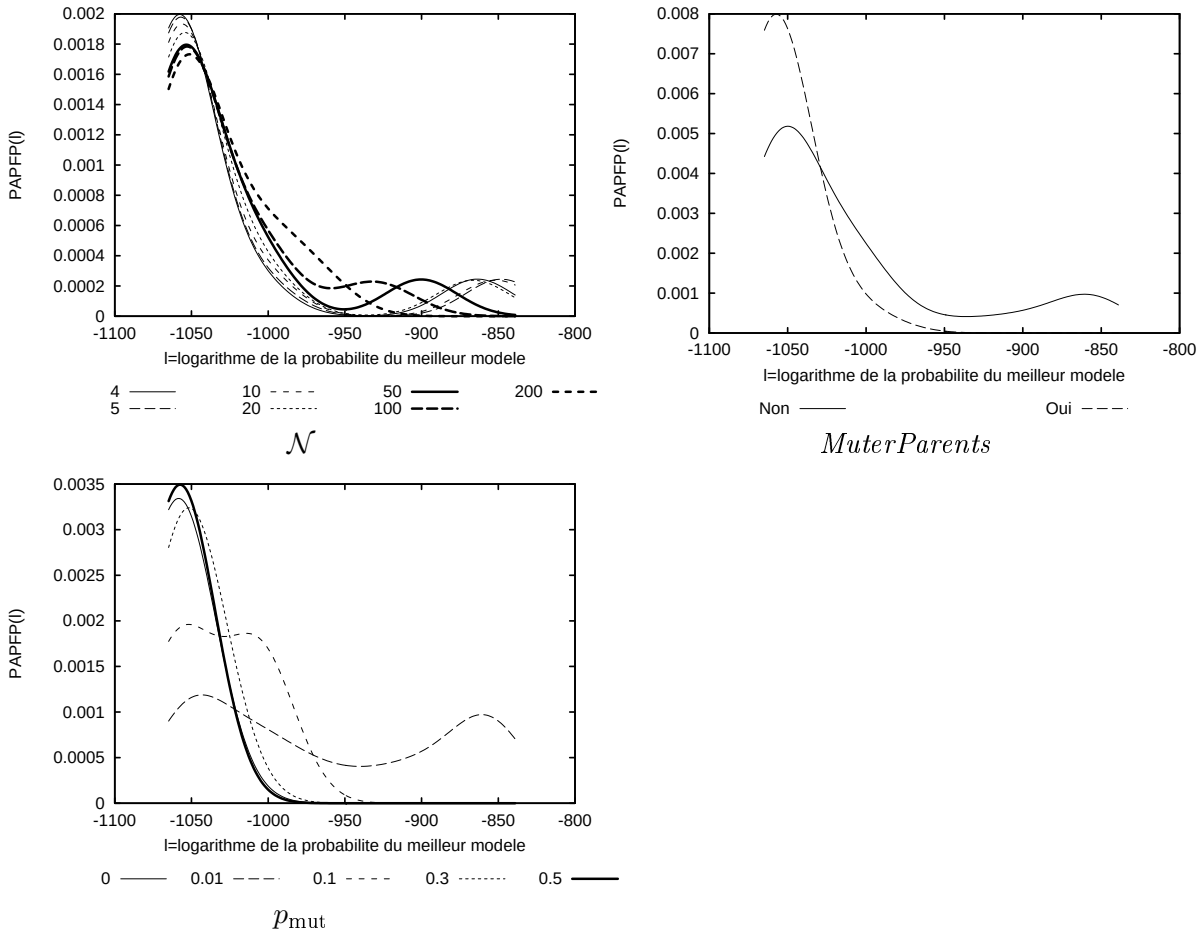


FIG. 4.26 – PAPFP de AGDiscret.

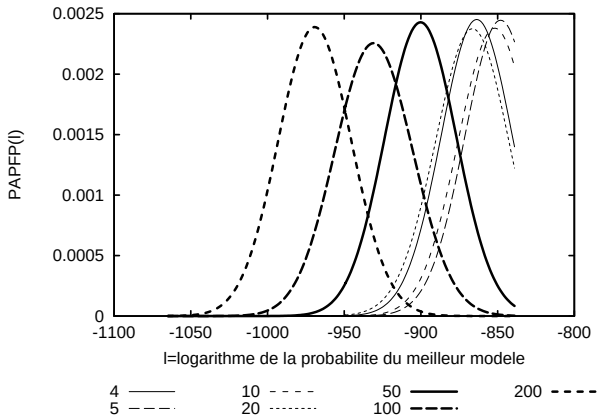


FIG. 4.27 – PAPFP de AGDiscret pour $\mathcal{N}$ lorsque *MuterParents*=Non et $p_{\text{mut}} = 0.01$.

4.5.1.9 La métaheuristique API*

Pour l'algorithme API*, nous avons considéré les mêmes paramètres que l'algorithme API (cf. table 4.4). Les deux types de configurations de paramètres se nomment APIHete* et APIHomo*.

TAB. 4.8 – Paramètres de l'algorithme APIDiscret.

APIDiscretHomo	
Paramètres	Domaines de valeurs
$\mathcal{N}$	2, 5, 10, 20, 50, 100
$\mathcal{T}_{\max}$	$1000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} > 0$ $30000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} = 0$
$\mathcal{M}_{\max}$	1
$e_{\max}$	1, 2, 3, 4, 5
$\mathcal{T}_{\text{Déplacement}}$	5, 10, 15, 20
$\mathcal{A}^i_{\text{site}}$	0.1, 0.2, 0.3, 0.4, 0.5
$\mathcal{A}^i_{\text{local}}$	0.1, 0.2, 0.3, 0.4, 0.5

APIDiscretHete	
Paramètres	Domaines de valeurs
$\mathcal{N}$	2, 5, 10, 20, 50, 100
$\mathcal{T}_{\max}$	$1000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} > 0$ $30000/\mathcal{N}$ si $\mathcal{N}_{\text{BW}} = 0$
$\mathcal{M}_{\max}$	1
$e_{\max}$	1, 2, 3, 4, 5
$\mathcal{T}_{\text{Déplacement}}$	5, 10, 15, 20
$\mathcal{A}^i_{\text{site}}$	$0.01(\frac{1}{0.01}^{\frac{i}{\mathcal{N}}})$
$\mathcal{A}^i_{\text{local}}$	$0.01(\frac{1}{0.01}^{\frac{i}{\mathcal{N}}})/10$

Etude des configurations de paramètres APIHomo*

La figure 4.31 présente les graphes PAPFP obtenus pour Img_1 lorsque $\mathcal{N}_{\text{BW}} = 0$. On remarque que les meilleures performances sont obtenues pour $\mathcal{N} = 2$ et qu'augmenter le nombre de fourmis diminue significativement les performances. Sous ces conditions, on montre que $e_{\max} = 1$, $e_{\max} = 5$, $\mathcal{A}^i_{\text{local}} = 0.9$ et $\mathcal{A}^i_{\text{site}} = 0.8$ mais les différences de performances sont relativement faibles.

La figure 4.32 présente les graphes PAPFP lorsque $\mathcal{N}_{\text{BW}} = 2$ obtenu pour Img_1. On remarque que les meilleures performances sont obtenues pour $\mathcal{N} = 20$. Comme précédemment, les autres paramètres ont peu d'influence sur les performances, néanmoins $\mathcal{T}_{\text{Déplacement}} = 5$, $e_{\max} = 4$, $\mathcal{A}^i_{\text{local}} = 0.1$ et $\mathcal{A}^i_{\text{site}} = 0.2$ semblent meilleurs.

La figure 4.33 présente les graphes PAPFP lorsque $\mathcal{N}_{\text{BW}} = 5$ obtenu pour Img_1. On remarque que les meilleures performances sont obtenues pour $\mathcal{N} = 50$. Les autres paramètres ont peu d'influence mais $\mathcal{T}_{\text{Déplacement}} = 15$, $e_{\max} = 4$, $\mathcal{A}^i_{\text{local}} = 0.1$ et $\mathcal{A}^i_{\text{site}} = 0.2$ semblent meilleurs.

Nous pouvons donc considérer les trois configurations de paramètres suivantes comme étant optimales :

- APIHomo1* : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 2$, $\mathcal{A}^i_{\text{local}} = 0.9$, $\mathcal{A}^i_{\text{site}} = 0.8$, $e_{\max} = 1$, $\mathcal{T}_{\text{Déplacement}} = 5$
- APIHomo2* : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 20$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.2$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 5$
- APIHomo3* : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 50$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.2$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$

On remarque que le nombre de fourmis $\mathcal{N}$ est le paramètre qui détermine les performances de l'algorithme. Les autres paramètres n'ont qu'une influence très limitée. On note également qu'augmenter le nombre d'itérations de l'algorithme Baum-Welch réduit le nombre d'essais que chacune des fourmis a besoin d'effectuer.

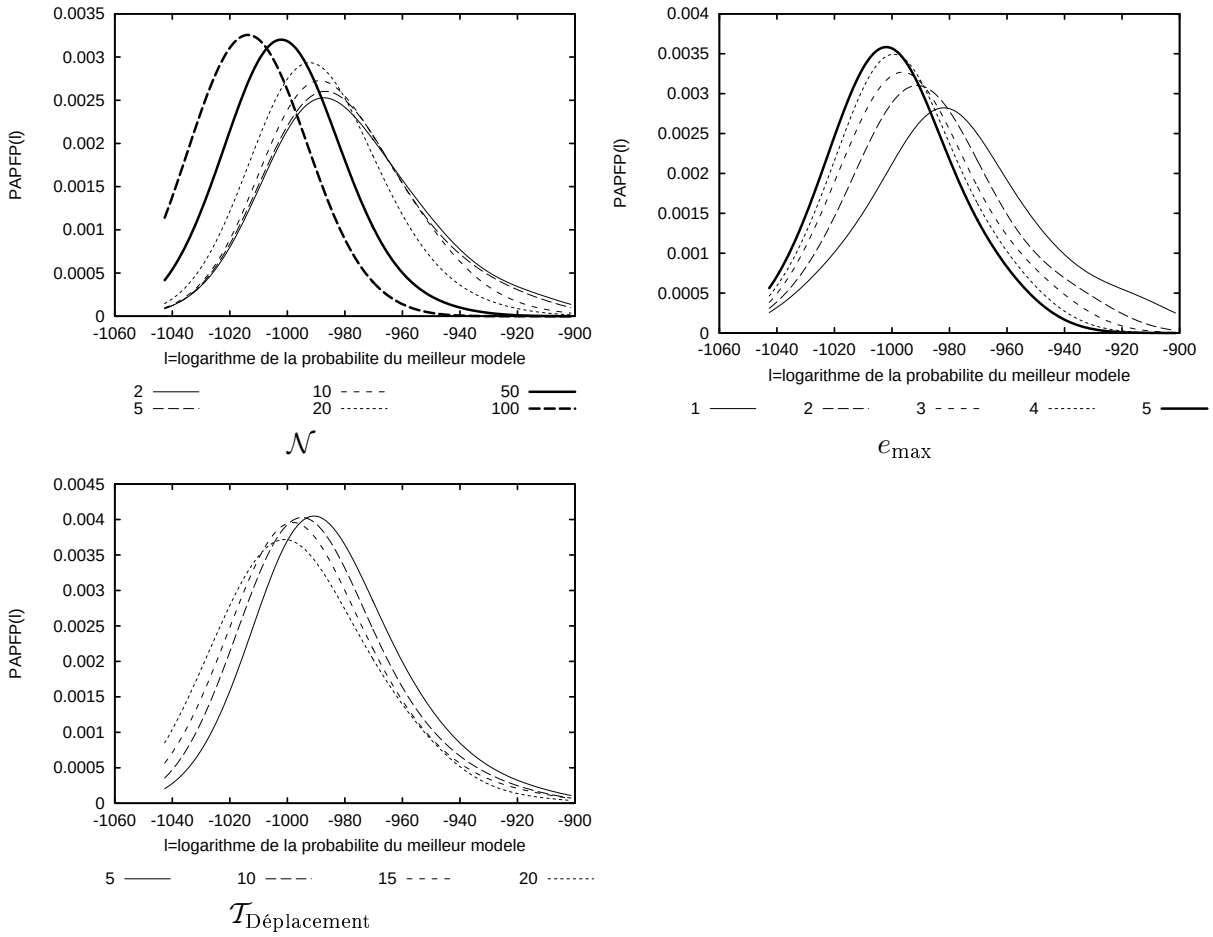


FIG. 4.28 – PAPFP de APIDiscretHomo.

Etude des configurations de paramètres APIHete*

La figure 4.34 présente les graphes PAPFP lorsque $\mathcal{N}_{\text{BW}} = 0$ obtenu pour Img_1. Nous remarquons que le paramètre le plus discriminant est $\mathcal{N}$: le nombre de fourmis. La valeur permettant d'obtenir la meilleure performance est $\mathcal{N} = 2$. De plus, augmenter cette valeur diminue les performances. Sous ces conditions, $\mathcal{T}_{\text{Déplacement}} = 5$ et $e_{\max} = 4$. Par conséquent, chaque fourmi a le temps d'exploiter entièrement sa mémoire et de la renouveler une fois avant que le nid ne soit déplacé.

La figure 4.35 présente les graphes PAPFP lorsque $\mathcal{N}_{\text{BW}} = 2$ obtenu pour Img_1. De même que précédemment, nous remarquons que le paramètre le plus significatif est $\mathcal{N}$, le nombre de fourmis, qui doit prendre pour valeur 20. On remarque également qu'une augmentation ou une diminution du nombre de fourmis diminue les performances et augmente les écart-types. Sous ces conditions, mais avec une influence très limitée, on a $\mathcal{T}_{\text{Déplacement}} = 5$ et $e_{\max} = 2$.

La figure 4.36 présente les graphes PAPFP lorsque $\mathcal{N}_{\text{BW}} = 5$ obtenu pour Img_1. De même que précédemment, nous remarquons que le paramètre le plus significatif est $\mathcal{N}$ qui doit prendre pour valeur 50. Sous ces conditions, mais avec une influence très limitée, on a $\mathcal{T}_{\text{Déplacement}} = 20$ et $e_{\max} = 1$.

Nous pouvons donc considérer les trois configurations de paramètres suivantes comme étant optimales :

- APIHete1* : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 2$, $e_{\max} = 5$, $\mathcal{T}_{\text{Déplacement}} = 4$
- APIHete2* : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 20$, $e_{\max} = 2$, $\mathcal{T}_{\text{Déplacement}} = 5$
- APIHete3* : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 50$, $e_{\max} = 1$, $\mathcal{T}_{\text{Déplacement}} = 20$

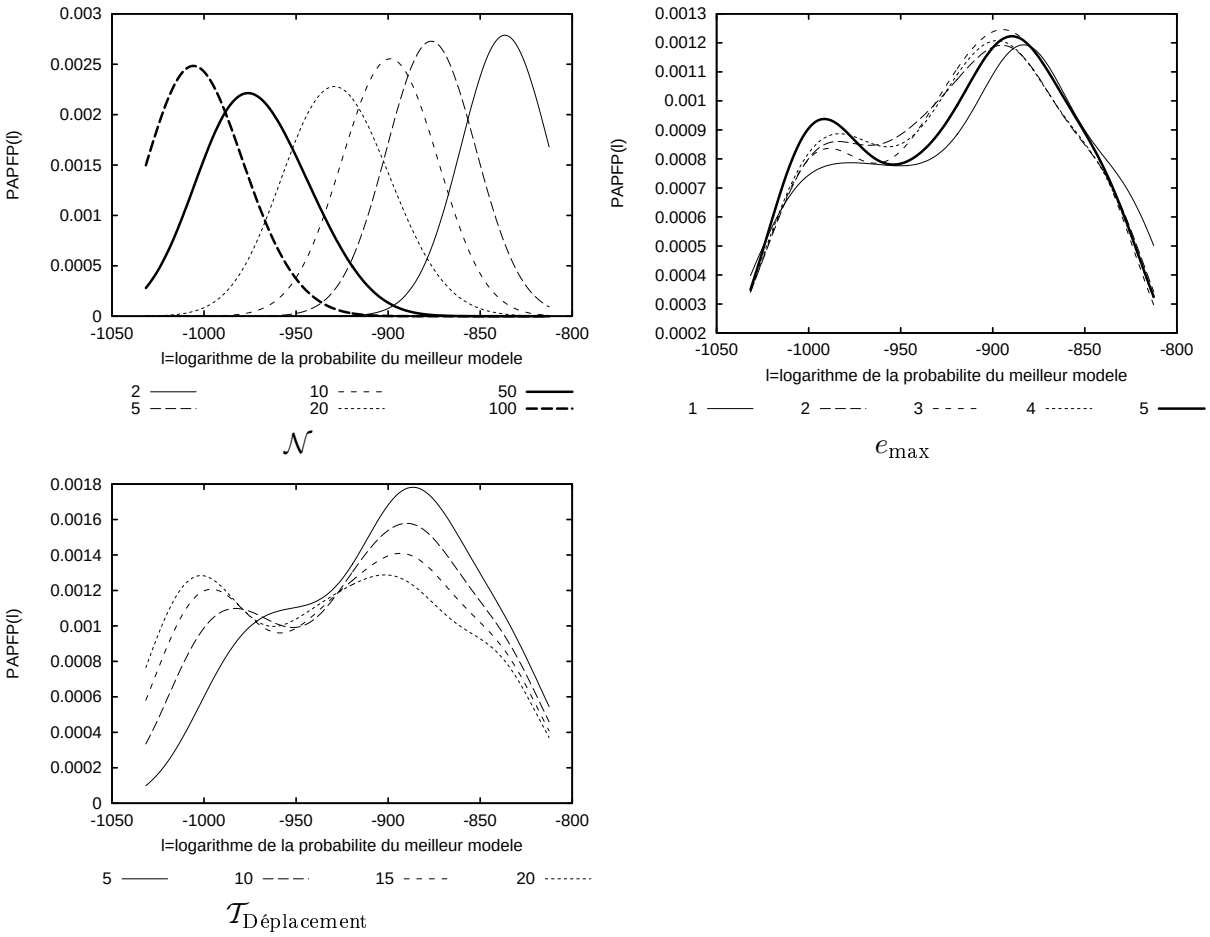


FIG. 4.29 – PAPFP de APIDiscretHete.

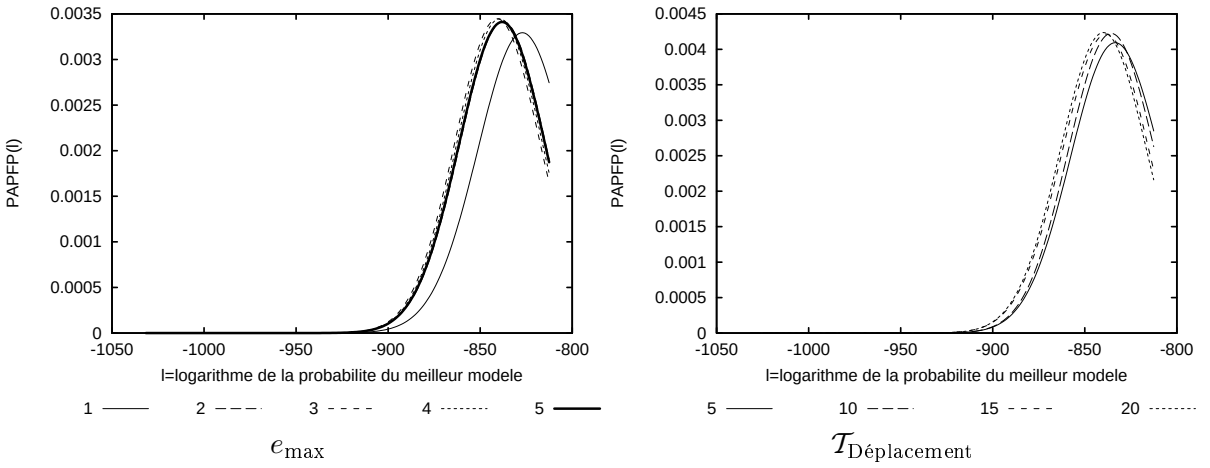


FIG. 4.30 – PAPFP de APIDiscretHete lorsque $\mathcal{N} = 2$.

On remarque que le nombre de fourmis $\mathcal{N}$ est le paramètre qui détermine les performances de l'algorithme. Les autres paramètres n'ont qu'une influence très limitée. On note également qu'augmenter le nombre d'itérations de l'algorithme de Baum-Welch réduit le nombre d'essais à réaliser par chacune des fourmis.

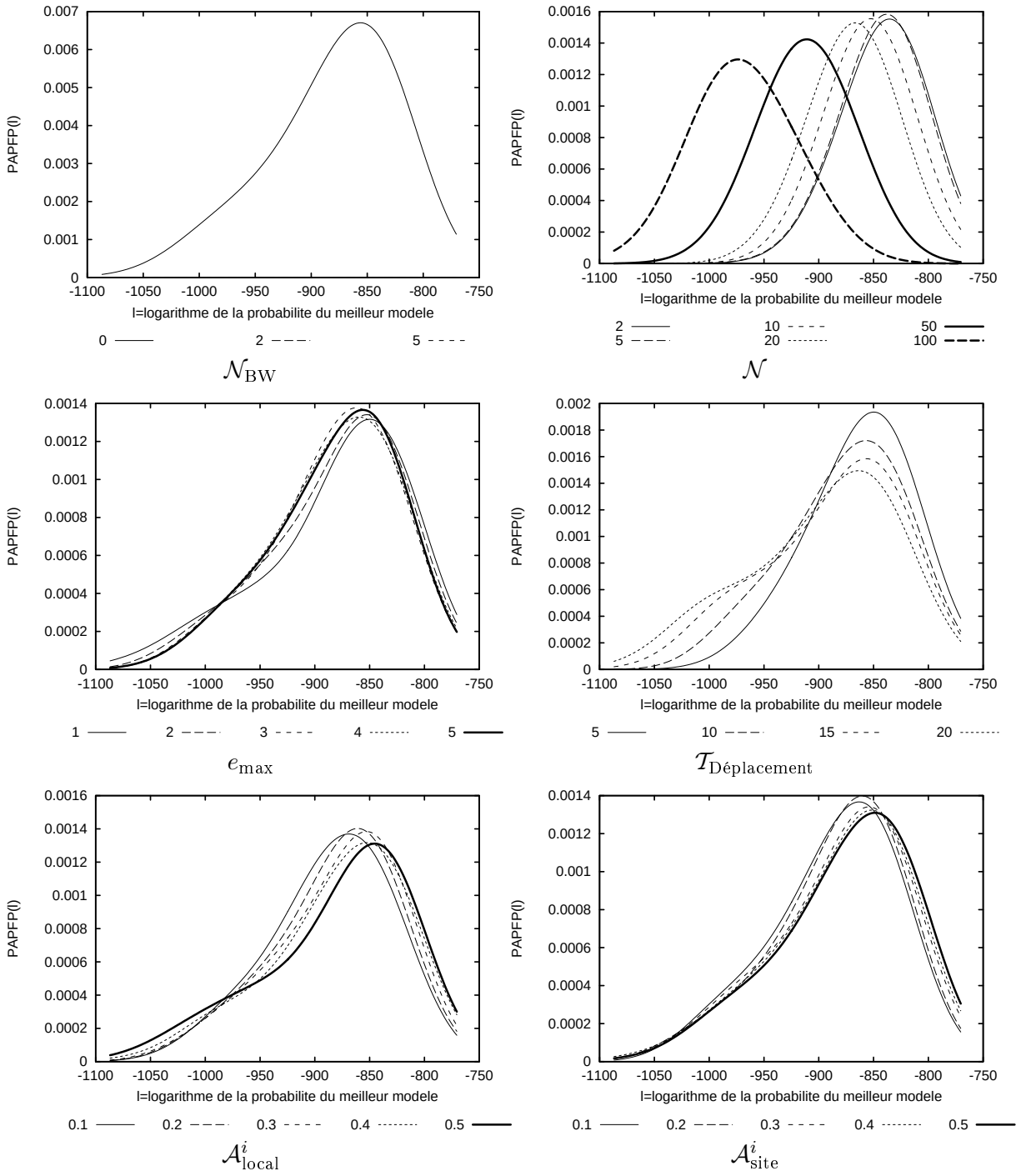


FIG. 4.31 – PAPFP de APIHomo* lorsque $\mathcal{N}_{\mathrm{BW}} = 0$.

4.5.2 Comparaison des performances

Pour évaluer les différentes méthodes les unes par rapport aux autres, nous avons construit les courbes moyennes de la probabilité du meilleur modèle trouvé en fonction :

- du nombre de modèles évalués *i.e.* du nombre de *Forward* lorsque l'algorithme de Baum-Welch n'est pas utilisé ;
- du nombre total d'itérations de Baum-Welch effectuées, lorsque l'algorithme de Baum-Welch est utilisé.

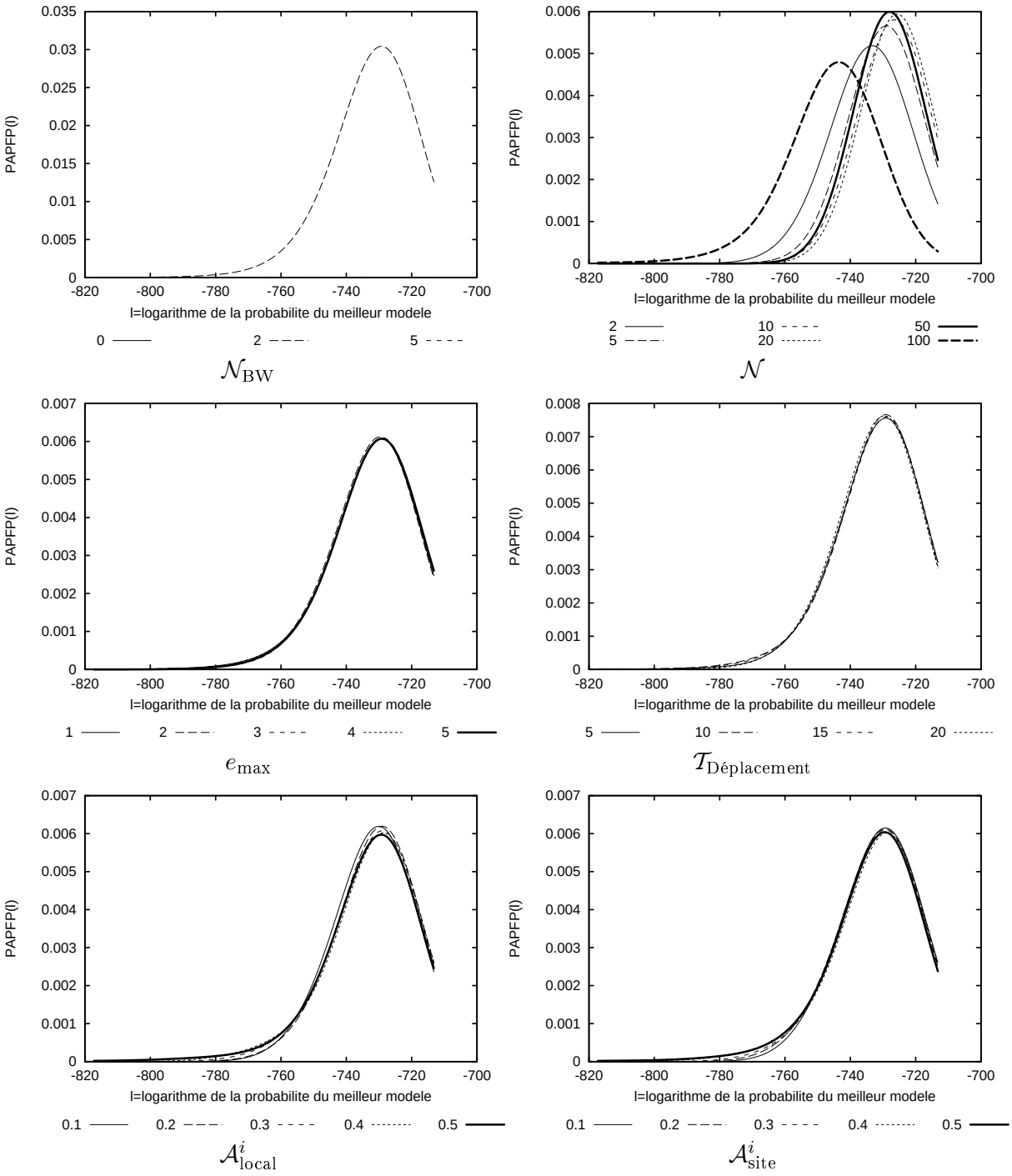


FIG. 4.32 – PAPFP de APIHomo* lorsque $\mathcal{N}_{\mathrm{BW}} = 2$.

Pour les tests, nous avons considéré les configurations de « bons » paramètres obtenues précédemment. Ces configurations sont :

- AG1 : $\mathcal{N}_{\mathrm{BW}} = 0$, *MuterParents*=Non, $p_{\mathrm{mut}} = 0.01$, $\mathcal{N} = 5$
- AG2 : $\mathcal{N}_{\mathrm{BW}} = 2$,$\mathcal{N} = 5$, $p_{\mathrm{mut}} = 0.3$, *MuterParents*=Oui et *OptimiserParents*=Oui
- AG3 : $\mathcal{N}_{\mathrm{BW}} = 2$,$\mathcal{N} = 20$, $p_{\mathrm{mut}} = 0.3$, *MuterParents*=Non et *OptimiserParents*=Oui
- AG4 : $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 20$, $p_{\mathrm{mut}} = 0.5$, *MuterParents*=Oui et *OptimiserParents*=Oui
- AG5 : $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 5$, $p_{\mathrm{mut}} = 0.5$, *MuterParents*=Non et *OptimiserParents*=Oui
- APIHomo1 : $\mathcal{N}_{\mathrm{BW}} = 0$, $\mathcal{N} = 50$, $\mathcal{A}^i_{\mathrm{local}} = 0.1$, $\mathcal{A}^i_{\mathrm{site}} = 0.1$, $e_{\max} = 1$, $\mathcal{T}_{\mathrm{Déplacement}} = 15$

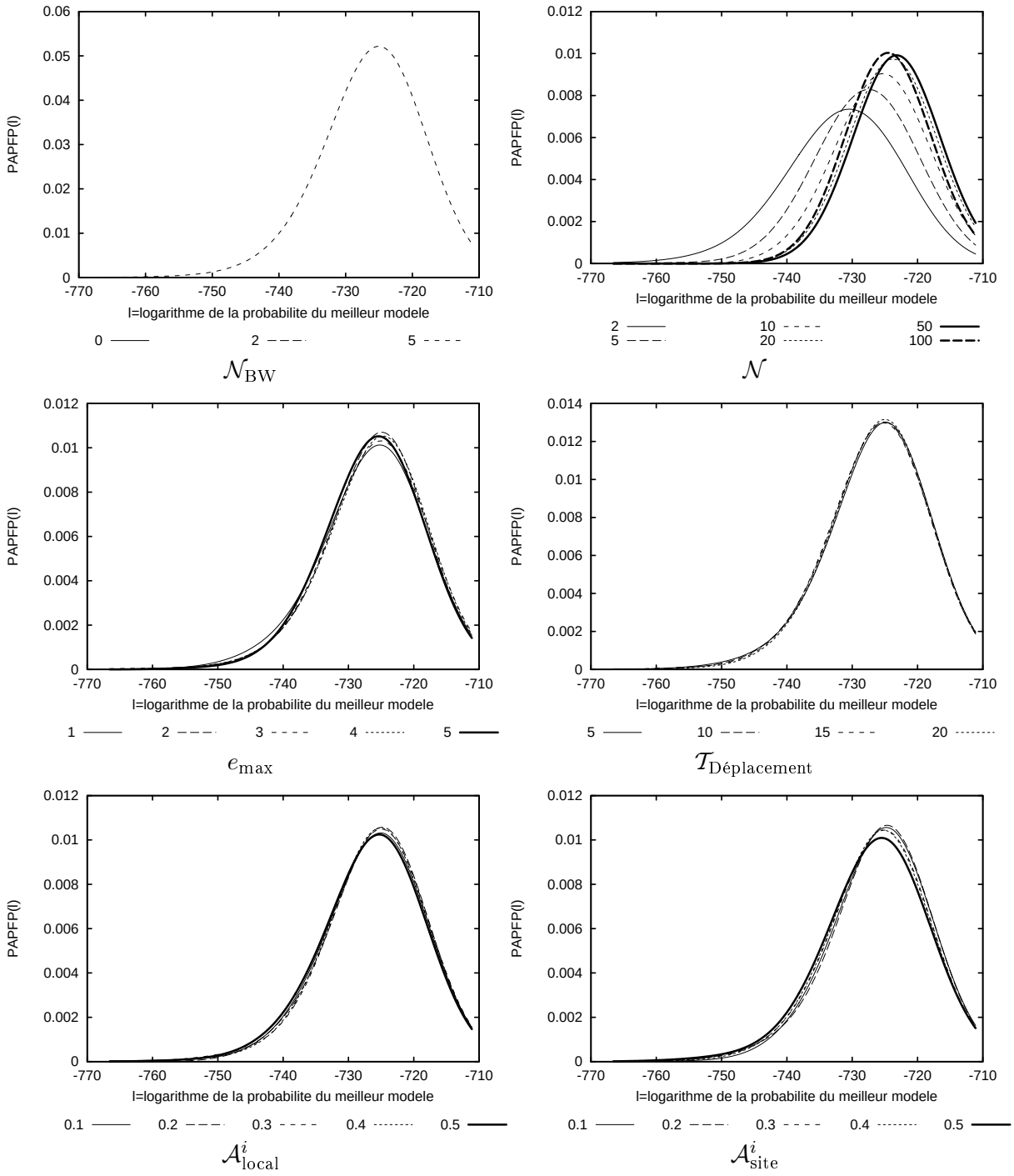


FIG. 4.33 – PAPFP de APIHomo* lorsque $\mathcal{N}_{\mathrm{BW}} = 5$.

- APIHomo2 : $\mathcal{N}_{\mathrm{BW}} = 2$, $\mathcal{N} = 5$, $\mathcal{A}^i_{\mathrm{local}} = 0.1$, $\mathcal{A}^i_{\mathrm{site}} = 0.1$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHomo3 : $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 20$, $\mathcal{A}^i_{\mathrm{local}} = 0.1$, $\mathcal{A}^i_{\mathrm{site}} = 0.1$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHete1 : $\mathcal{N}_{\mathrm{BW}} = 0$, $\mathcal{N} = 2$, $e_{\max} = 5$, $\mathcal{T}_{\text{Déplacement}} = 10$
- APIHete2 : $\mathcal{N}_{\mathrm{BW}} = 2$, $\mathcal{N} = 5$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHete3 : $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 5$, $e_{\max} = 3$, $\mathcal{T}_{\text{Déplacement}} = 5$
- OEPDistance1 : $\mathcal{N}_{\mathrm{BW}} = 0$, $\mathcal{N} = 100$, $\omega = 1$, $c_1 = 0.5$, $c_2 = 0.5$, $d = d_0$
- OEPDistance2 : $\mathcal{N}_{\mathrm{BW}} = 2$, $\mathcal{N} = 20$, $\omega = 0.4$, $c_1 = 1$, $c_2 = 0$
- OEPDistance3 : $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 50$, $\omega = 0.2$, $c_1 = 0$, $c_2 = 0$, $V = 6$

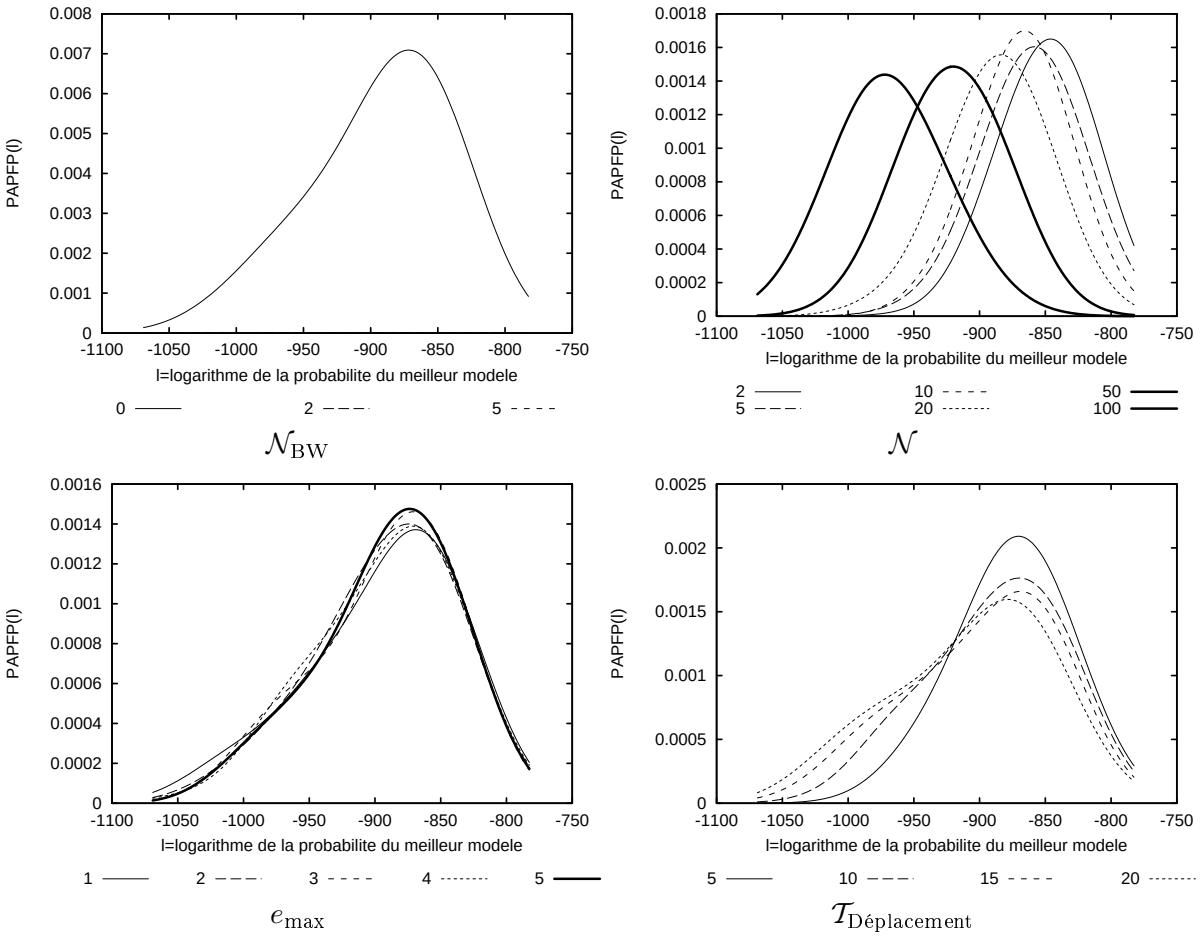


FIG. 4.34 – PAPFP de APIHete* lorsque $\mathcal{N}_{\mathrm{BW}} = 0$.

- OEPSocial1 : $\mathcal{N}_{\mathrm{BW}} = 0$, $\mathcal{N} = 20$, $\omega = 0.8$, $c_1 = 1.5$, $c_2 = 2.5$, $V = 9$
- OEPSocial2 : $\mathcal{N}_{\mathrm{BW}} = 2$, $\mathcal{N} = 20$, $\omega = 0.8$, $c_1 = 1.5$, $c_2 = 0$
- OEPSocial3 : $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 50$, $\omega = 0.4$, $c_1 = 0$, $c_2 = 0.5$, $V = 6$
- AGDiscret *MuterParents*=Non, $p_{\mathrm{mut}} = 0.01$, $\mathcal{N} = 5$
- APIDiscretHomo : $e_{\max} = 1$, $\mathcal{N} = 2$, $\mathcal{T}_{\mathrm{Déplacement}} = 5$, $\mathcal{A}^i_{\mathrm{site}} = 0.1$ et $\mathcal{A}^i_{\mathrm{local}} = 0.2$.
- APIDiscretHete : $e_{\max} = 1$, $\mathcal{N} = 2$, $\mathcal{T}_{\mathrm{Déplacement}} = 5$
- APIHomo1* : $\mathcal{N}_{\mathrm{BW}} = 0$, $\mathcal{N} = 2$, $\mathcal{A}^i_{\mathrm{local}} = 0.9$, $\mathcal{A}^i_{\mathrm{site}} = 0.8$, $e_{\max} = 1$, $\mathcal{T}_{\mathrm{Déplacement}} = 5$
- APIHomo2* : $\mathcal{N}_{\mathrm{BW}} = 2$, $\mathcal{N} = 20$, $\mathcal{A}^i_{\mathrm{local}} = 0.1$, $\mathcal{A}^i_{\mathrm{site}} = 0.2$, $e_{\max} = 4$, $\mathcal{T}_{\mathrm{Déplacement}} = 5$
- APIHomo3* : $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 50$, $\mathcal{A}^i_{\mathrm{local}} = 0.1$, $\mathcal{A}^i_{\mathrm{site}} = 0.2$, $e_{\max} = 4$, $\mathcal{T}_{\mathrm{Déplacement}} = 15$
- APIHete1* : $\mathcal{N}_{\mathrm{BW}} = 0$, $\mathcal{N} = 2$, $e_{\max} = 5$, $\mathcal{T}_{\mathrm{Déplacement}} = 4$
- APIHete2* : $\mathcal{N}_{\mathrm{BW}} = 2$, $\mathcal{N} = 20$, $e_{\max} = 2$, $\mathcal{T}_{\mathrm{Déplacement}} = 5$
- APIHete3* : $\mathcal{N}_{\mathrm{BW}} = 5$, $\mathcal{N} = 50$, $e_{\max} = 1$, $\mathcal{T}_{\mathrm{Déplacement}} = 20$

Une heuristique a été ajoutée à ces expérimentations, afin de déterminer si les métaheuristiques apportent réellement quelque chose par rapport à une approche plus simple. La première forme de l'heuristique, que nous nommerons Random0, consiste à engendrer des modèles aléatoirement dans Λ et à ne conserver que le meilleur modèle trouvé. La deuxième et troisième forme, nommées respectivement Random2 et Random5, appliquent 2 ou 5 itérations de Baum-Welch à chacun des modèles engendrés aléatoirement, avant de considérer leur vraisemblance.

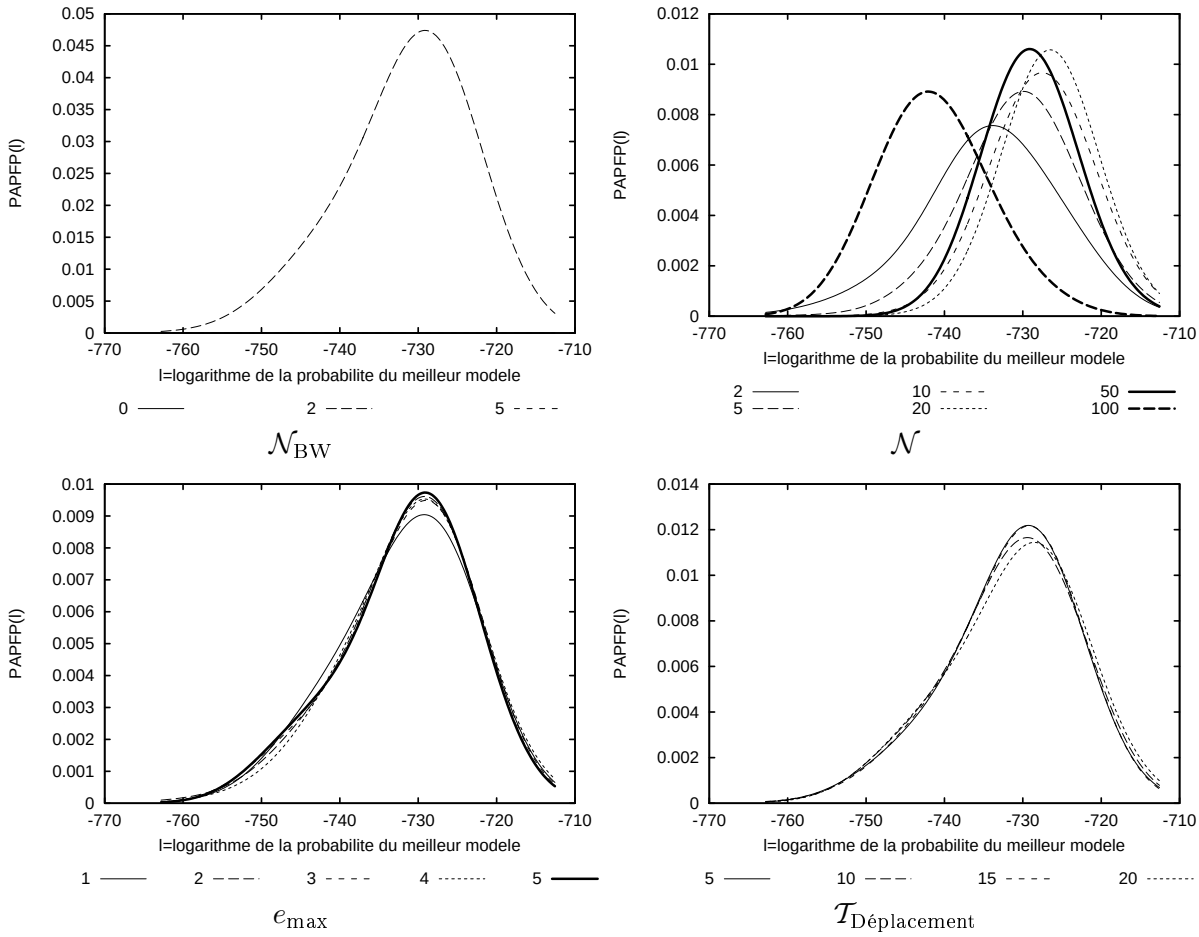


FIG. 4.35 – PAPFP de APIHete* lorsque $\mathcal{N}_{\mathrm{BW}} = 2$.

Les séquences d'observations considérées pour ces comparaisons sont obtenues à partir des quatre images utilisées précédemment, mais en faisant varier le nombre de symboles lors du ré-échelonnement *i.e.* M et le nombre d'états cachés dans les modèles *i.e.* N. Ces différentes combinaisons nous permettent de comparer les algorithmes sur des espaces Λ (ou Ω ou *Lambda**) de taille différente. Les graphiques résultants sont donnés par les figures D.1, D.2, D.3, D.4, D.5, D.6, D.7, D.8, D.9, D.10, D.11 et D.12 situées en annexe D.

4.5.2.1 Le cas $\mathcal{N}_{\mathrm{BW}} = 0$

Lorsque l'algorithme de Baum-Welch n'est pas utilisé, on remarque que trois groupes se forment. Le premier groupe, correspondant aux algorithmes les plus performants, est composé de AG1, APIHomo1*, AGDiscret, APIDiscretHete, APIDiscretHomo et APIHete*. On remarque que AG1, suivi de APIHomo1* et APIHete*, atteint globalement les meilleures performances. Le deuxième groupe d'algorithmes correspond à PSODistance1, PSOSocial1, Random0, APIHete1. Finalement, on remarque que APIHomo1 forme le groupe qui atteint les plus mauvaises performances.

On remarque que les paramétrages homogènes (APIHomo1, APDiscretHomo, APIHomo*) permettent d'atteindre des performances significativement inférieures à celles des paramétrages hétérogènes associés (APIHete1, APIDiscretHete, APIHete*). Ces résultats confirment les remarques faites dans (Monmarché, 2000) indiquant qu'un paramétrage hétérogène de API est plus efficace qu'un paramétrage homogène de API.

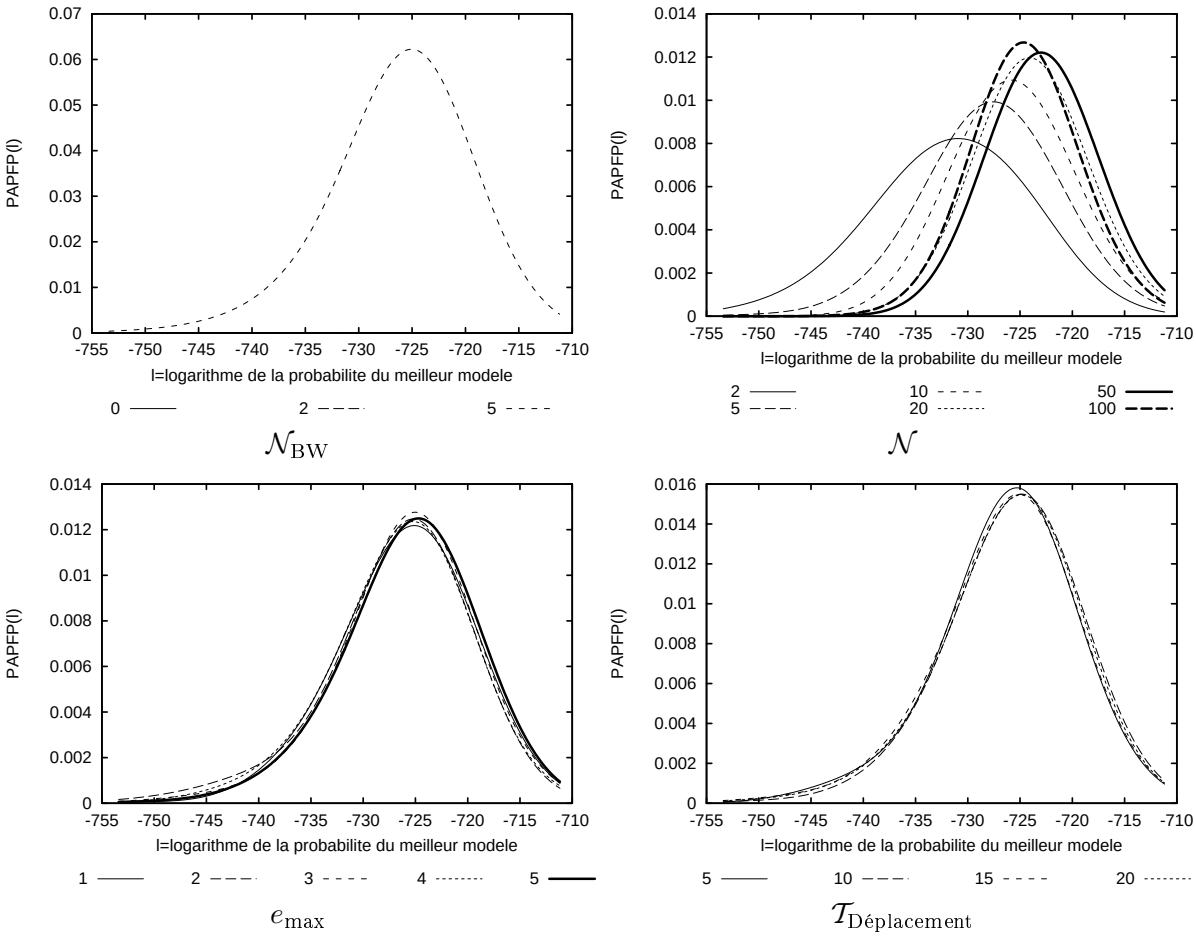


FIG. 4.36 – PAPFP de APIHete* lorsque $\mathcal{N}_{\mathrm{BW}} = 5$.

On remarque que APIHomo est moins efficace que Random0. Ce comportement peut paraître assez déroutant au premier abord, mais il semblerait qu'une recherche aléatoire ait plus de chance de trouver un meilleur optimum qu'une recherche « mal guidée ».

Un phénomène intéressant qui méritera d'être exploré, et confirmé dans de futurs travaux, est le fait que les algorithmes travaillant sur $\mathcal{S}^T$ démarrent leur recherche avec des modèles significativement meilleurs que les autres modèles. L'apprentissage étiqueté de modèle à partir de séquences d'états aléatoires pourrait être un moyen efficace d'initialiser les positions initiales des autres algorithmes.

4.5.2.2 Le cas $\mathcal{N}_{\mathbf{BW}} = 2$

Lorsque deux itérations de Baum-Welch sont utilisées, trois groupes d'algorithmes apparaissent. Le premier groupe, obtenant les meilleures performances, est composé de APIHete2 et AG2. Le deuxième groupe est composé de AG3, PSOSocial2, PSODistance2, APIHomo2* et APIHete2*. Le troisième groupe, correspondant aux plus mauvaises performances, est composé de Random2 et APIHomo2.

On remarque que, dans la quasi totalité des cas, APIHete2 atteint des performances supérieures ou égales à celle de AG2. De plus, les algorithmes formant le deuxième groupe atteignent des performances équivalentes. Finalement, on remarque que APIHomo2 atteint des performances très en dessous des autres algorithmes, mais globalement, elles restent supérieures à la recherche aléatoire Random2.

4.5.2.3 Le cas $\mathcal{N}_{\mathbf{BW}} = 5$

Lorsque cinq itérations de Baum-Welch sont utilisées, on distingue trois groupes d'algorithmes. Le premier groupe, correspondant aux performances les plus élevées, est composé de APIHete3, APIHomo3 et AG4. On remarque que globalement ce groupe est dominé par APIHete3, mais les performances de APIHomo3 et AG4 sont très proches. Le deuxième groupe est composé de AG5, PSOSocial3, PSODistance3, APIHomo3* et APIHete3*. Les performances des algorithmes de ce groupe sont similaires. Cependant, AG5 a le mérite de converger beaucoup plus rapidement que les autres algorithmes.

4.5.2.4 Comparaison des meilleurs algorithmes

Dans un premier temps, nous remarquons que les performances des meilleurs algorithmes n'utilisant pas de l'algorithme de Baum-Welch sont significativement inférieures à celles des meilleurs algorithmes l'utilisant. De plus, pour obtenir ces performances nettement inférieures, il a été nécessaire d'évaluer beaucoup de modèles. Or, ce nombre est tel que, comparativement à l'utilisation de l'algorithme de Baum-Welch, les temps d'exécution sont plus longs. Nous ne donnons pas les temps d'éxecution car les différentes implémentations ne bénéficient pas toutes du même niveau d'optimisation. Cependant, on peut remarquer que $M > N$ donc la complexité de l'algorithme de Baum-Welch est $O(NMT)$. Or, réaliser 30000 évaluations de l'algorithme *Forward* équivaut (en ordre de grandeur) à $30000TN^2$ instructions tandis que 1000 évaluations de Baum-Welch équivaut (en ordre de grandeur) à $1000NMT$ instructions. Le rapport *Forward*/Baum-Welch est (en ordre de grandeur) $30N/M$. Ce rapport prend des valeurs comprises entre 5 et 18 pour nos expérimentations. Ce rapport étant nettement supérieur à 1, les algorithmes sans l'algorithme de Baum-Welch nécessitent plus de temps de calcul. Par conséquent, l'approche sans Baum-Welch n'est pas intéressante.

Lorsque Baum-Welch est utilisé, on remarque que deux types d'algorithmes se détâchent des autres en terme de performances : AG (AG2, AG4) et API (APIHete2, APIHete3) avec un paramétrage hétérogène. Pour étudier ces différents algorithmes, nous avons défini Random10, Random100 et Random1000 de manière identique à Random2 ou Random5 mais avec 10, 100 et 1000 itérations de Baum-Welch. Les graphes résultants sont donnés par les figures D.13, D.14, D.15 et D.16 situées en annexe. On remarque que APIHete2 converge plus rapidement que APIHete5 vers des vraisemblances de même qualité, donc il n'est pas forcément utile d'effectuer beaucoup d'itérations de Baum-Welch, contrairement à ce que l'on pourrait croire au premier abord. La même remarque peut être faite concernant l'algorithme génétique.

Les différentes recherches aléatoires (Random2, Random5, Random10, Random100, Random1000) montrent que l'utilisation de métaheuristiques est nécessaire pour trouver les meilleurs modèles. Les graphes montrent également qu'augmenter le nombre d'itérations de Baum-Welch dans une recherche aléatoire améliore, dans un premier temps, les performances et qu'au delà d'un certain nombre, l'effort d'optimisation réalisé par l'algorithme de Baum-Welch est vain.

Finalement, on remarque que API, avec ses paramétrages hétérogènes, atteint des vraisemblances supérieures ou équivalentes à l'algorithme génétique ou à une recherche aléatoire, aussi bien lorsque $\mathcal{N}_{\mathrm{BW}} = 2$ que lorsque $\mathcal{N}_{\mathrm{BW}} = 5$. Ces résultats confirment les expériences menées dans (Monmarché, 2000).

4.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'adaptation et à l'évaluation de métaheuristiques à base de populations pour l'apprentissage de MMC. Plusieurs métaheuristiques

utilisant des espaces de recherche différents ont été proposées et comparées. Une structure originale d'espace vectoriel a pu être définie pour les MMC permettant d'envisager de nouvelles méthodes d'optimisation, telles que l'optimisation par essaim particulaire. Nous avons montré que les algorithmes génétiques et l'algorithme API offraient un bon niveau de performance supérieur à notre adaptation de l'OEP. De nombreuses améliorations restent à envisager pour l'OEP, tels que les coefficients de constriction ou la définition de bornes pour l'amplitude de la vitesse, avant de pouvoir considérer l'OEP sur cet espace vectoriel comme étant moins performante que les autres algorithmes sur ce type de problème.

L'adaptation de API et de l'algorithme génétique à l'espace $\mathbb{S}^T$ a révélé que l'apprentissage étiqueté avec une séquence d'états choisie aléatoirement permettait de considérer des MMC initiaux nettement meilleurs en terme de performance que les MMC choisis uniformément dans Λ. Grâce à ces propriétés, nous pouvons conclure qu'il est intéressant d'envisager différents espaces de solutions lors de l'apprentissage de MMC, car ceux-ci ont des propriétés différentes pouvant être des avantages ou des inconvénients.

Les algorithmes ont été comparés sur l'apprentissage de MMC pour le critère de maximum de vraisemblance. Afin d'en faire de véritables standards pour l'apprentissage de MMC, il nous reste à les évaluer sur d'autres jeux de données, sur d'autres critères et sur d'autres types de modèles de Markov cachés, tels que les modèles de Markov cachés à Substitution de Symboles (cf. chapitre 5).

Si l'on se réfère à la table 4.2, on s'aperçoit que plusieurs possibilités restent à explorer. L'OEP est depuis peu employée pour la résolution de problèmes discrets (Kennedy and Eberhart, 1997) (Rameshkumar et al., 2005). Il pourrait donc être intéressant de considérer ces variantes de l'OEP originale et d'évaluer l'OEP discrète sur l'espace $\mathbb{S}^T$. De même, l'algorithme génétique pourrait tirer parti de la structure vectorielle définie sur Ω afin d'effectuer l'exploration de l'espace des solutions. Ces adaptations feront très probablement partie de nos recherches futures.

Finalement, nos travaux ont mis en évidence que les espaces discrets $\mathbb{S}^T$ sont moins propices à une bonne exploration de l'espace. Cette limite est principalement due au fait que l'on ne peut pas utiliser d'algorithme d'optimisation locale tel que celui de Baum-Welch. Cependant, nous avons vu qu'ils permettaient d'obtenir très rapidement des modèles « moyennement bons ». De plus, si l'on s'autorisait à perdre une partie de la cohérence globale des algorithmes en utilisant une optimisation locale, on montrerait peut-être que cet espace est très efficace pour la recherche de modèles optimaux. Ces expérimentations feront l'objet de recherches futures.

Concernant les espaces Ω ou Λ^* munis d'une structure vectorielle, il nous semble prématuré de conclure. En effet, les algorithmes les utilisant donnent des résultats inférieurs, mais nous ne savons pas si cela provient de la structure de l'espace ou des opérateurs considérés.

L'espace Λ a été celui le plus anciennement utilisé et, au vu de nos expérimentations, il risque de l'être encore longtemps, avec succès, pour l'apprentissage des modèles.

Chapitre 5

Modèles de Markov cachés à substitutions de symboles (MMCSS)

Les modèles de Markov cachés à substitutions de symboles (MMCSS) sont un nouveau type de MMC (Aupetit et al., 2002). Ces modèles ont été créés par S. Aupetit, N. Monmarché et M. Slimane (Aupetit, 2002), pour répondre à un objectif simple : permettre l'incorporation de connaissances *a priori* pendant l'apprentissage et la reconnaissance d'images. A cet effet, nous avons choisi d'incorporer une couche de variables aléatoires intermédiaires entre les états cachés et les symboles émis. Dans la suite, nous les nommerons « méta symboles intermédiaires ».

Le principe dissimulé derrière l'appellation MMCSS est simple. On considère un MMC discret ordinaire, comme ceux décrits au chapitre 1, sauf que la génération des symboles observés s'effectue différemment. Les états cachés émettent un « symbole » dit intermédiaire qui est transformé en un symbole émis (ou observé), à l'aide d'une loi de substitution définie *a priori*.

Dans ce chapitre, nous développons les principes et la théorie qui régissent les modèles de Markov cachés à substitutions de symboles. Depuis la création des MMCSS, ces modèles ont donné lieu à l'encadrement de plusieurs mini-projets (Hublier and Téboul, 2003) (Benkirane and Bouyahyaoui, 2003) (Lesné and Perotin, 2004a) (Peiti, 2005). De plus, les MMCSS ont été utilisés dans plusieurs travaux non liés à cette thèse. On peut notamment noter les travaux réalisés par deux projets de fin d'études (Vidal, 2004) (Martin, 2005) dans le cadre de la thèse de Thierry Hénocque sur les systèmes de détection d'intrusion dans les réseaux.

5.1 Définition

Définition 13 *Soit $\mathbb{S} = \{s_1, \ldots, s_N\}$ l'ensemble des N états cachés du système. Soit $S = (S_1, \ldots, S_T)$ un T-uple de v.a. définies sur $\mathbb{S}$. Soit $\mathbb{V} = \{v_1, \ldots, v_M\}$ l'ensemble des M symboles émissibles par le système. Soient $I = (I_1, \ldots, I_T)$ et $V = (V_1, \ldots, V_T)$ deux T-uples de v.a. définies sur $\mathbb{V}$.*

Un modèle de Markov caché discret à substitutions de symboles est alors défini par les probabilités suivantes :

- *les probabilités d'initialisation des états cachés : $P(S_1 = s_i)$*
- *les probabilités de transition entre états cachés : $P(S_t = s_j / S_{t-1} = s_i)$*
- *les probabilités d'émission des symboles intermédiaires dans chaque état caché : $P(I_t = v_j / S_t = s_i)$*

De plus, on note $P(V_t = v_j / I_t = v_i)$ les probabilités de substitution a priori entre les symboles (transformation des symboles intermédiaires en symboles émis).

Dans le cas d'un modèle de Markov caché à substitutions de symboles stationnaire et du premier ordre, on peut définir les matrices $A = (a_{i,j})_{1\leq i,j\leq N}$, $B = (b_i(j))_{1\leq i\leq N, 1\leq j\leq M}$ et $\Pi = (\pi_1, \ldots, \pi_N)'$ en notant $a_{i,j} = P(S_t = s_j / S_{t-1} = s_i)$, $b_i(j) = P(I_t = v_j / S_t = s_i)$ et $\pi_i = P(S_1 = s_i)$ pour $t > 1$ quelconque. Un modèle de Markov caché à substitutions de symboles stationnaire du premier ordre λ est donc totalement défini par le triplet (A, B, Π). Par la suite, nous utiliserons la notation $\lambda = (A, B, \Pi)$ et nous emploierons le terme MMCSS pour désigner ces modèles particuliers.

Dans la suite, nous supposons également que la loi de substitution est stationnaire dans le temps. On a donc $P(V_t = v_j / I_t = v_i) = P(V_{t'} = v_j / I_{t'} = v_i)$ quels que soient les instants t et t'. On définit $c_i(j) = P(V_t = v_j / I_t = v_i)$. La loi de substitution est donc totalement définie par la matrice stochastique $C = (c_i(j))_{1\leq i,j\leq N}$. Le lecteur notera que cette substitution pourrait faire partie du modèle mais étant donné qu'elle est définie *a priori*, il nous paraît plus adéquat de la considérer séparément, voire de la considérer au même titre que la séquence d'observations. Dans la suite, C sera omis lorsqu'il n'y aura pas d'ambiguïté.

Les relations de dépendance entre les différentes variables aléatoires d'un MMCSS sont schématisées par la figure 5.1. Dans cette représentation, les flèches débutent de la v.a. qui conditionne et se terminent au niveau de la variable aléatoire conditionnée. Dans la figure, seules les transitions au temps $t-1$, t et $t+1$ sont représentées.

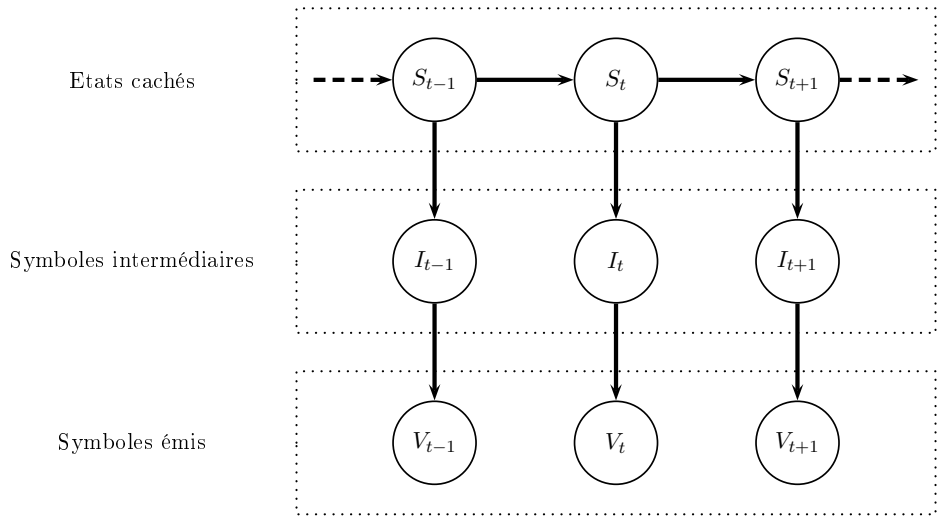


FIG. 5.1 – Relations de dépendance entre les variables aléatoires d'un MMCSS.

Soit $O \in \mathbb{V}^T$ une séquence d'observations de longueur T, $J \in \mathbb{V}^T$ une séquence de symboles intermédiaires et $Q \in \mathbb{S}^T$ une séquence d'états cachés, alors, en utilisant les dépendances des probabilités conditionnelles, on remarque que

$$P(V = O, I = J, S = Q/\lambda) = P(V = O/I = J, \lambda) P(I = J/S = Q, \lambda) P(S = Q/\lambda)$$

Or la loi de substitution étant indépendante des paramètres du modèle, on a

$$P(V = O/I = J, \lambda) = P(V = O/I = J)$$

d'où

$$P(V = O, I = J, S = Q/\lambda) = P(V = O/I = J) P(I = J/S = Q, \lambda) P(S = Q/\lambda)$$

De plus,

$$P(V = O/I = J) = \prod_{t=1}^{T} P(V_t = o_t / I_t = j_t)$$

$$P(I = J/S = Q, \lambda) = \prod_{t=1}^{T} P(I_t = o_t/S_t = q_t, \lambda)$$

$$P(S = Q/\lambda) = P(S_1 = q_1/\lambda) \prod_{t=1}^{T-1} P(S_{t+1} = q_{t+1}/S_t = q_t, \lambda)$$

La vraisemblance de l'observation O par rapport au modèle λ est donnée par

$$P(V = O/\lambda) = \sum_{J \in \mathbb{V}^T} \sum_{Q \in \mathbb{S}^T} P(V = O, I = J, S = Q/\lambda)$$

On peut dès lors remarquer que les MMCSS ne sont qu'un cas très particulier des modèles de Markov cachés hiérarchiques (Fine et al., 1998). En effet, les MMC hiérarchiques généralisent de nombreux types de MMC discrets et les MMCSS ne font pas exception. On pourrait alors se demander l'intérêt de les considérer distincts de ces derniers. L'intérêt est double : ils sont plus simples à manipuler et la définition d'algorithmes spécialisés permet d'effectuer les calculs plus efficacement, notamment en exploitant le fait que la loi de substitution est connue *a priori*. On peut également remarquer que lorsque la loi de substitution est donnée par $P(V_t = o_t/I_t = o_t) = 1$ (et par conséquent 0 ailleurs) alors le MMCSS est strictement identique à un MMC. Par conséquent, le pouvoir d'expression d'un MMCSS est supérieur à celui d'un MMC (pour une loi de substitution bien choisie).

5.2 Incorporation de connaissances expertes via la loi de substitution

Comme nous l'avons dit plus haut, l'intérêt principal des MMCSS est de permettre l'incorporation de connaissances *a priori* dans l'apprentissage et la reconnaissance des séquences d'observations. Pour cela, il suffit de choisir une bonne loi. Étant donné que les symboles intermédiaires sont en même nombre que les symboles émis, il devient naturel de considérer les symboles intermédiaires comme des « méta » symboles correspondant à la modélisation du processus. Les symboles émis peuvent alors être considérés comme des symboles « dégénérés », dans le sens où ils devraient correspondre aux « méta » symboles associés avec une certaine confiance. Ainsi, à partir d'une séquence d'observations, le MMCSS se comporte comme s'il remplaçait chaque symbole émis par les « méta » symboles, tout en évaluant la confiance dans cette substitution sous la forme de probabilités.

Deux exemples d'ajout de connaissances *a priori* :

- Les symboles sont des niveaux de gris (NdG) : lorsqu'on effectue l'apprentissage d'images en NdG à l'aide d'un MMC, une difficulté se pose. Il arrive très souvent que le nombre de NdG différents apparaissant dans une image soit plus réduit que le nombre total de niveaux de gris autorisé. Dans de tels cas, l'apprentissage du MMC à partir d'une image peut aboutir à des probabilités nulles d'émission des NdG non présents dans l'image. Considérons l'exemple d'une image I et d'un MMC λ, obtenu par l'apprentissage de l'image I. Considérons également une image I', identique à I mais pour laquelle un pixel, choisi au hasard, est remplacé par un NdG n'apparaissant pas dans l'image I. Alors, la vraisemblance de I' par rapport au MMC λ est nulle. Or, les deux images sont identiques à un pixel près. La modification d'un seul élément de l'image a rendu les deux images différentes. Ce comportement est, en pratique, peu souhaitable. Plusieurs approches peuvent être utilisées pour pallier à ce problème. L'une d'elles consiste a ré-échelonner les NdG de manière à obtenir moins de NdG et de ce fait réduire les risques précédents. L'inconvénient d'un ré-échelonnement est qu'il peut rendre très différents deux NdG similaires. Par conséquent, ce n'est pas toujours une bonne solution.

Une solution alternative consiste à prendre en compte, à l'apprentissage et à la reconnaissance, la notion de NdG similaires. Les MMCSS permettent de le faire grâce à la loi de substitution. Pour cela, il suffit, pour chaque niveau de gris x et chaque NdG $y \neq x$, de prendre $c_{x,x} > c_{x,y}$. La définition des $c_{x,y}$ peut prendre différentes formes, telles qu'une gaussienne, une loi triangle, ... centrée sur le NdG x. Ces lois permettent d'assurer qu'un méta-symbole x sera plus probablement remplacé par le symbole x que par un autre symbole.

– Les angles sur un disque : lorsqu'on cherche à apprendre l'évolution de directions (Nord, Nord-Est, Est, Sud-Est, Sud, Sud-Ouest, Ouest, Nord-Ouest), il n'est pas adéquat de coder la séquence d'observations selon les 8 directions. En effet, la limite entre, par exemple, Nord et Nord-Est n'est pas stricte. Un changement d'angle de 1 degré est-il vraiment significatif pour différencier Nord et Nord-Est ? De plus, une direction à la limite de ces deux directions risque de produire une séquence d'observations oscillant de manière aléatoire entre ces deux directions. Du point de vue du MMC, la modélisation de ce processus risque d'être difficile. Une solution consiste à incorporer la notion de similarité des directions sur le disque. On peut utiliser, par exemple, $c_{x,x} = 0.5$ et $c_{x,x-1} = 0.25$ et $c_{x,x+1} = 0.5$ avec x la « méta » direction, $x-1$ et $x+1$ les deux directions adjacentes.

5.3 Algorithmes principaux

Maintenant que nous savons ce que sont les MMCSS et comment il est possible d'incorporer des connaissances *a priori* dans l'apprentissage et la reconnaissance, il est nécessaire de donner les algorithmes associés aux MMCSS permettant de résoudre les problèmes de calcul de la vraisemblance, de décodage/segmentation de séquences d'observations et de l'apprentissage.

Dans plusieurs des algorithmes suivants, il est possible de réduire la complexité des algorithmes si l'on suppose que le nombre de symboles du modèle est inférieur à la longueur de la séquence d'observations ($M < T$). Dans ce cas, il est possible d'effectuer le pré-calcul $f_j(k) = \sum_{v=1}^{M} b_j(v)c_v(k)$ pour tout $j = 1..N$ et $k = 1..M$. Ce calcul possède une complexité en $O(NM^2)$. Ce pré-calcul permet de réduire la complexité des algorithmes suivants, au détriment de l'occupation mémoire, qui nécessite dès lors NM cases mémoires supplémentaires. Cependant, dans le cas où $M > T$, la complexité est augmentée, car certains calculs sont effectués alors qu'ils ne sont pas utiles. Dans ce cas, il est nécessaire de ne pas effectuer de pré-calcul et d'effectuer le calcul de ces valeurs uniquement lorsqu'on en a besoin. Les complexités exprimées ci-après partiront de l'hypothèse $M < T$ mais, dans la pratique, on préférera calculer les $f_j(k)$ lorsqu'on en aura besoin, ceci permettant de garantir un temps de calcul quasi minimum dans tous les cas, mais au détriment de l'utilisation mémoire. Cette stratégie a été adoptée lors de l'implémentation de ces algorithmes dans la bibliothèque HMMTK (cf. chapitre 7).

5.3.1 Calcul de la vraisemblance

5.3.1.1 Algorithme *Forward*

Pour présenter rapidement cet algorithme, il est nécessaire de définir les variables *Forward* (pour tout $i = 1..N$ et $t > 1$) :

$$\begin{cases} \alpha_1(i) = P(V_1 = o_1, S_1 = s_i/\lambda) \\ \alpha_t(i) = P(V_1 = o_1, \ldots, V_t = o_t, S_t = s_i/\lambda) \end{cases}$$

On remarque alors que la relation de récurrence

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{i,j}\right) \left(\sum_{v=1}^{M} b_j(v) c_v(o_{t+1})\right)$$

est vérifiée pour tout $t = 1..T-1$ et $j = 1..N$. De plus, on a $P(V = O/\lambda) = \sum_{i=1}^{N} \alpha_T(i)$. L'algorithme *Forward* est alors donné par l'algorithme 5.1. La complexité de cet algorithme est en $O(NM^2 + N^2T)$. On remarque alors que, si on ne considère pas le pré-calcul $f_j(k)$, la complexité est identique à celle de l'algorithme *Forward* des MMC.

Algorithme 5.1: *Forward* MMCSS

Pour $j = 1$ **à** N **Faire**
  Pour $k = 1$ **à** M **Faire**
    $f_j(k) = \sum_{v=1}^{M} b_j(v) c_v(k)$
  Fin Pour
Fin Pour
Pour $i = 1$ **à** N **Faire**
  $\alpha_1(i) = \pi_i f_i(o_1)$
Fin Pour
Pour $t = 1$ **à** $T-1$ **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) f_j(o_{t+1})$
  Fin Pour
Fin Pour
$P(V = O/\lambda) = \sum_{i=1}^{N} \alpha_T(i)$

Comme pour les MMC, il est nécessaire de définir une version avec ré-échelonnement de cet algorithme. Les notations et les principes généraux décrits dans la section 1.4.1 sont utilisés afin d'obtenir l'algorithme 5.2. La complexité de cet algorithme reste identique à celle de l'algorithme 5.1.

5.3.1.2 Algorithme *Backward*

Pour l'algorithme de ré-estimation des paramètres d'un modèle, il est nécessaire de définir l'algorithme *Backward* associé. Cet algorithme est donné par l'algorithme 5.3. Une alternative utilisant le ré-échelonnement est donnée par l'algorithme 5.4. Comme précédemment, les notations introduites à la section 1.4.1 sont utilisées. La complexité de ces deux algorithmes est $O(NM^2 + TN^2)$.

5.3.1.3 Probabilités déductibles

A partir des variables *Forward* et *Backward*, avec ou sans ré-échelonnement, il nous est d'ores et déjà possible d'exprimer trois probabilités utiles dans la section suivante.

$$\begin{aligned} P(V = O, S_t = s_i/\lambda) &= \alpha_t(i)\beta_t(i) \\ &= \frac{\tilde{\alpha}_t(i)\tilde{\beta}_t(i)}{P(V = O/\lambda)} \qquad (5.1) \\ P(V = O, S_t = s_i, S_{t+1} = s_j/\lambda) &= \alpha_t(i) a_{i,j} f_j(o_{t+1}) \beta_{t+1}(j) \\ &= \frac{\tilde{\alpha}_t(i) a_{i,j} f_j(o_{t+1}) \tilde{\beta}_{t+1}(j)}{P(V = O/\lambda)} \qquad (5.2) \\ P(V = O, I_t = v_j, S_t = s_i/\lambda) &= \alpha_t(i,j)\beta_t(i) \qquad (5.3) \end{aligned}$$

Algorithme 5.2: *Forward* MMCSS avec ré-échelonnement

Pour $j = 1$ **à** N **Faire**
  Pour $k = 1$ **à** M **Faire**
    $f_j(k) = \sum_{v=1}^{M} b_j(v)c_v(k)$
  Fin Pour
Fin Pour
Pour $i = 1$ **à** N **Faire**
  $\alpha'_1(i) = \pi_i f_i(o_1)$
Fin Pour
$c_1 = \frac{1}{\sum_{i=1}^{N} \alpha'_1(i)}$
Pour $i = 1$ **à** N **Faire**
  $\tilde{\alpha}_1(i) = c_1\alpha'_1(i)$
Fin Pour
Pour $t = 1$ **à** $T-1$ **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\alpha'_{t+1}(j) = \left(\sum_{i=1}^{N} \tilde{\alpha}_t(i)a_{ij}\right) f_j(o_{t+1})$
  Fin Pour
  $c_t = \frac{1}{\sum_{j=1}^{N} \alpha'_t(j)}$
  Pour $j = 1$ **à** N **Faire**
    $\tilde{\alpha}_{t+1}(j) = c_t\alpha'_{t+1}(j)$
  Fin Pour
Fin Pour
$P(V = O/\lambda) = \frac{1}{\prod_{t=1}^{T} c_t}$

avec

$$\alpha_t(i,j) = \begin{cases} \pi_i b_i(j)c_j(o_1) & \text{si } t = 1 \\ \left(\sum_{k=1}^{N} \alpha_{t-1}(k)a_{k,i}\right) b_i(j)c_j(o_t) & \text{sinon} \end{cases}$$

On remarque que ce calcul peut être effectué efficacement à partir des variables que l'on connaît déjà, grâce à la formule suivante $\alpha_t(i,j) = \frac{\alpha_t(i)}{f_i(o_t)} b_i(j)c_j(o_t)$. On a donc

$$\begin{aligned} P(V = O, I_t = v_j, S_t = s_i/\lambda) &= \alpha_t(i)\frac{b_i(j)c_j(o_t)}{f_i(o_t)}\beta_t(i) \\ &= \tilde{\alpha}_t(i)\frac{b_i(j)c_j(o_t)}{f_i(o_t)P(V = O/\lambda)}\tilde{\beta}_t(i) \end{aligned}$$

5.3.2 Ré-estimation des paramètres

5.3.2.1 Maximisation de la vraisemblance

Pour effectuer l'apprentissage d'un MMCSS sous l'hypothèse de maximisation de la vraisemblance, on cherche à maximiser $P(V = O/\lambda)$ avec O une séquence de T observations. En appliquant l'algorithme EM (cf. section 1.6.2.1) à la maximisation de cette probabilité, on est amené à maximiser $\Gamma(\lambda, \lambda')$ en désignant par $\lambda = (\Pi, A, B)$ le nouveau modèle et λ' le modèle connu (ou actuel) :

$$\Gamma_O(\lambda, \lambda') = \sum_{J \in \mathbb{V}^T} \sum_{Q \in \mathbb{S}^T} P(I = J, S = Q/V = O, \lambda') \ln P(V = O, I = J, S = Q/\lambda)$$

Algorithme 5.3: *Backward* MMCSS

Pour $j = 1$ **à** N **Faire**
  Pour $k = 1$ **à** M **Faire**
    $f_j(k) = \sum_{v=1}^{M} b_j(v) c_v(k)$
  Fin Pour
Fin Pour
Pour $i = 1$ **à** N **Faire**
  $\beta_T(i) = 1$
Fin Pour
Pour $t = T-1$ **à** 1 **Faire**
  Pour $i = 1$ **à** N **Faire**
    $\beta_t(i) = \sum_{j=1}^{N} a_{ij} \beta_{t+1}(j) f_j(o_{t+1})$
  Fin Pour
Fin Pour
$P(V = O/\lambda) = \sum_{i=1}^{N} \pi_i f_i(o_1) \beta_1(i)$

Algorithme 5.4: *Backward* MMCSS avec ré-échelonnement

Pour $j = 1$ **à** N **Faire**
  Pour $k = 1$ **à** M **Faire**
    $f_j(k) = \sum_{v=1}^{M} b_j(v) c_v(k)$
  Fin Pour
Fin Pour
Pour $i = 1$ **à** N **Faire**
  $\tilde{\beta}_T(i) = c_T$
Fin Pour
Pour $t = T-1$ **à** 1 **Faire**
  Pour $i = 1$ **à** N **Faire**
    $\tilde{\beta}_t(i) = c_t \sum_{j=1}^{N} a_{i,j} \tilde{\beta}_{t+1}(j) f_j(o_{t+1})$
  Fin Pour
Fin Pour

Sachant que

$$P(V = O, I = J, S = Q/\lambda) = \pi_{q_1} \left(\prod_{t=1}^{T-1} a_{q_t q_{t+1}} \right) \left(\prod_{t=1}^{T} c_{j_t}(o_t) \right) \left(\prod_{t=1}^{T} b_{q_t}(j_t) \right)$$

la fonction Γ_O se ré-écrit

$$\begin{aligned}
\Gamma_O(\lambda, \lambda') &= \sum_{Q \in \mathbb{S}^T} \ln \pi_{q_1} P(S = Q/V = O, \lambda') \\
&+ \sum_{Q \in \mathbb{S}^T} \left(\sum_{t=1}^{T-1} \ln a_{q_t q_{t+1}} \right) P(S = Q/V = O, \lambda') \\
&+ \sum_{J \in \mathbb{V}^T} \sum_{Q \in \mathbb{S}^T} \left(\sum_{t=1}^{T} \ln b_{q_t}(j_t) \right) P(I = J, S = Q/V = O, \lambda') \\
&+ \sum_{J \in \mathbb{V}^T} \left(\sum_{t=1}^{T} \ln c_{j_t}(o_t) \right) P(I = J/V = O, \lambda') \\
&= \Gamma_O^{\pi}(\lambda, \lambda') + \Gamma_O^{A}(\lambda, \lambda') + \Gamma_O^{B}(\lambda, \lambda') + \Gamma_O^{C}(\lambda, \lambda')
\end{aligned}$$

On peut alors effectuer plusieurs remarques :
- $\Gamma(\lambda, \lambda')$ se décompose en la somme de quatre fonctions de paramètres distincts et indépendants, par conséquent il est possible de les maximiser indépendamment les unes des autres.
- $\Gamma_O^C(\lambda, \lambda')$ ne dépend pas de λ, donc nous pouvons ignorer ce terme dans la maximisation de Γ_O.
- L'expression des termes $\Gamma_O^\Pi(\lambda, \lambda')$ et $\Gamma_O^A(\lambda, \lambda')$ est identique à celle qu'on obtient en appliquant EM à l'apprentissage de MMC, donc les formules de ré-estimation suivantes se déduisent facilement :

$$\begin{aligned} \pi_i &= P(S_1 = s_i / V = O, \lambda') \\ a_{i,j} &= \frac{\sum_{t=1}^{T-1} P(S_t = s_i, S_{t+1} = s_j / V = O, \lambda')}{\sum_{t=1}^{T-1} P(S_t = s_i / V = O, \lambda')} \end{aligned}$$

Il suffit donc d'exprimer la ré-estimation des coefficients $b_i(j)$. Pour cela, il suffit de remarquer que l'expression $\Gamma_O^B(\lambda, \lambda')$ se simplifie :

$$\begin{aligned} \Gamma_O^B(\lambda, \lambda') &= \sum_{J \in \mathbb{V}^T} \sum_{Q \in \mathbb{S}^T} \left(\sum_{t=1}^{T} \ln b_{q_t}(j_t) \right) P(I = J, S = Q / V = O, \lambda') \\ &= \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{j=1}^{M} \ln b_i(j) P(I_t = v_j, S_t = s_i / V = O, \lambda') \end{aligned}$$

En utilisant les multiplicateurs de Lagrange pour contraindre $\sum_{j=1}^{M} b_i(j) = 1$ et en dérivant, on obtient

$$\begin{aligned} &\frac{\partial}{\partial b_i(j)} \left(\Gamma_O^B(\lambda, \lambda') + \sum_{n=1}^{N} \gamma_n \left(\sum_{k=1}^{N} b_n(k) - 1 \right) \right) \\ &= \sum_{t=1}^{T} \frac{P(I_t = v_j, S_t = s_i / V = O, \lambda')}{b_i(j)} + \gamma_i \\ &= 0 \end{aligned}$$

et

$$\begin{aligned} \frac{\partial}{\partial \gamma_i} \left(\Gamma_O^B(\lambda, \lambda') + \sum_{n=1}^{N} \gamma_n \left(\sum_{k=1}^{N} b_n(k) - 1 \right) \right) &= \sum_{k=1}^{N} b_i(k) - 1 \\ &= 0 \end{aligned}$$

Alors en multipliant par $b_i(j)$ la première équation et en sommant sur j, on obtient, grâce à la deuxième équation

$$\gamma_i = -\sum_{t=1}^{T} P(S_t = s_i / V = O, \lambda')$$

et donc

$$b_i(j) = \frac{\sum_{t=1}^{T} P(I_t = v_j, S_t = s_i / V = O, \lambda')}{\sum_{t=1}^{T} P(S_t = s_i / V = O, \lambda')}$$

L'algorithme de Baum-Welch est donné par l'algorithme 5.5. Sa complexité est $O(NM^2 + N^2T + NMT)$.

Algorithme 5.5: Algorithme de Baum-Welch MMCSS

Choisir un modèle initial λ_0
$t = 0$
Répéter
  $t \leftarrow t + 1$
  Calculer les variables *Forward* et *Backward* pour le modèle λ_{t-1}
  Calculer Π de λ_t
  Calculer A de λ_t
  Calculer B de λ_t
Tant que $(P(V = O/\lambda_t) > P(V = O/\lambda_{t-1}))$ **et** $(t < t_{\max})$

D'une manière naïve, les probabilités utilisées pour la ré-estimation des matrices peuvent être obtenues par les algorithmes *Forward* et *Backward*. Cependant, toujours pour des problèmes d'implémentation numérique, on utilise plutôt leurs versions utilisant les algorithmes avec ré-échelonnement (cf. équations de la section 5.3.1.3).

5.3.2.2 Descente de gradient

Lorsque l'apprentissage consiste à optimiser un autre critère que la vraisemblance, il est parfois nécessaire d'avoir recours au gradient de la probabilité $P(V = O/\lambda)$. Dans cette section, nous allons développer le gradient de $P(V = O/\lambda)$ en reprenant les notations introduites à la section 1.6.2.3.

Un certain nombre de propriétés restent inchangées, ou se démontrent facilement :

$$L(\lambda) = \ln P(V = O/\lambda)$$

$$\frac{\partial L(\lambda)}{\partial \lambda}(\mu) = \frac{1}{P(V = O/\mu)} \frac{\partial P(V = O/\lambda)}{\partial \lambda}(\mu)$$

$$\begin{aligned}
\frac{\partial P(V = O/\lambda)}{\partial x_{i,j}} &= \sum_{t=2}^{T} \alpha_{t-1}(i) a_{i,j} f_j(o_t) \beta_t(j) - a_{i,j} \sum_{t=2}^{T} \alpha_{t-1}(i) \beta_{t-1}(i) \\
\frac{\partial P(V = O/\lambda)}{\partial z_i} &= \pi_i f_i(o_1) \beta_1(i) - \pi_i P(V = O/\lambda)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial P(V = O/\lambda)}{\partial y_{i,j}} &= \sum_{t=1}^{T} \sum_{l=1}^{N} \sum_{m=1}^{N} \frac{\partial P(V = O/\lambda)}{\partial \alpha_t(l)} . \frac{\partial \alpha_t(l)}{\partial b_l(o_t = m)} . \frac{\partial b_l(o_t = m)}{\partial y_{i,j}} \\
&= \sum_{t=1}^{T} \sum_{m=1}^{N} \frac{\partial P(V = O/\lambda)}{\partial \alpha_t(i)} . \frac{\partial \alpha_t(i)}{\partial b_i(o_t = m)} . \frac{\partial b_i(o_t = m)}{\partial y_{i,j}} \\
&= \sum_{t=1}^{T} \sum_{m=1}^{N} \frac{\partial P(V = O/\lambda)}{\partial \alpha_t(i)} . \frac{\partial \alpha_t(i)}{\partial b_i(m)} . \frac{\partial b_i(m)}{\partial y_{i,j}}
\end{aligned}$$

$$\alpha_t(i) = \begin{cases} \pi_i f_i(o_1) & \text{si } t = 1 \\ \sum_{k=1}^{N} \alpha_{t-1}(k) a_{k,i} f_i(o_t) & \text{si } t > 1 \end{cases}$$

donc

$$\begin{aligned}\frac{\partial \alpha_t(i)}{\partial b_i(m)} &= \begin{cases} \pi_i c_m(o_1) & \text{si } t=1 \\ \sum_{k=1}^{N} \alpha_{t-1}(j) a_{j,i} c_m(o_t) & \text{si } t>1 \end{cases} \\ &= \alpha_t(i) \frac{b_i(m) c_m(o_t)}{f_i(o_t)}\end{aligned}$$

d'où

$$\begin{aligned}\frac{\partial P(V=O/\lambda)}{\partial y_{i,j}} &= \sum_{t=1}^{T} \sum_{m=1}^{N} \beta_t(i) \alpha_t(i) \frac{b_i(m) c_m(o_t)}{f_i(o_t)} (\kappa(m=j) - b_i(j)) \\ &= \sum_{t=1}^{T} \beta_t(i) \frac{b_i(j) c_j(o_t)}{f_i(o_t)} \alpha_t(i) - b_i(j) \sum_{t=1}^{T} \beta_t(i) \alpha_t(i)\end{aligned}$$

En utilisant les variables *Forward* et *Backward* avec ré-échelonnement, on obtient

$$\begin{aligned}\frac{1}{P(V=O/\lambda)} \frac{\partial P(V=O/\lambda)}{\partial x_{i,j}} &= \sum_{t=2}^{T} \tilde{\alpha}_{t-1}(i) a_{i,j} f_j(o_t) \tilde{\beta}_t(j) - a_{i,j} \sum_{t=2}^{T} \frac{\tilde{\alpha}_{t-1}(i) \tilde{\beta}_{t-1}(i)}{c_{t-1}} \\ \frac{1}{P(V=O/\lambda)} \frac{\partial P(V=O/\lambda)}{\partial y_{i,j}} &= \sum_{t=1}^{T} \beta_t(i) \frac{b_i(j) c_j(o_t)}{f_i(o_t)} \alpha_t(i) - b_i(j) \sum_{t=1}^{T} \beta_t(i) \alpha_t(i) \\ \frac{1}{P(V=O/\lambda)} \frac{\partial P(V=O/\lambda)}{\partial z_i} &= \pi_i f_i(o_1) \tilde{\beta}_1(i) - \pi_i\end{aligned}$$

A partir de ce gradient, il est possible d'utiliser n'importe quelle descente de gradient telle que celle décrite dans (Ganapathiraju, 1999) et (Kapadia, 1998). Cependant, il faut garder à l'esprit que ce calcul est coûteux en temps machine. Sa complexité est $O(N^2T + NMT)$.

5.3.3 Décodage/segmentation de séquences d'observations

Le décodage ou la segmentation de séquences d'observations (selon le même esprit que l'algorithme de Viterbi) peut être effectué de deux façons : soit en cherchant la séquence Q^* d'états cachés qui maximise la probabilité $P(V=O, S=Q^*/\lambda)$, soit en cherchant la séquence Q^* d'états cachés et la séquence J^* de symboles intermédiaires maximisant la probabilité $P(V=O, I=J^*, S=Q^*/\lambda)$. Dans cette section, nous allons montrer comment ces séquences peuvent être obtenues de manière efficace.

5.3.3.1 Recherche du chemin d'états cachés optimal

La recherche du chemin Q^* d'états cachés optimal se déduit facilement de l'algorithme *Forward* décrit ci-dessus et de l'algorithme de Viterbi décrit au chapitre 1. Il suffit juste de remplacer $b_i(j)$ par $f_i(j)$. Les notations utilisées pour les MMC sont conservées et les algorithmes de Viterbi sans et avec ré-échelonnement sont donnés par 5.6 et 5.7. La complexité de ces algorithmes est $O(NM^2 + TN^2)$.

5.3.3.2 Recherche des chemins d'états cachés et de symboles intermédiaires optimaux

Rechercher les chemins Q^* d'états cachés et J^* de symboles intermédiaires optimaux consiste à trouver Q^* et J^* maximisant $P(V=O, I=J^*, S=Q^*/\lambda)$. Dans cette optique,

Algorithme 5.6: Algorithme de Viterbi MMCSS

Pour $j = 1$ **à** N **Faire**
  Pour $k = 1$ **à** M **Faire**
    $f_j(k) = \sum_{v=1}^{M} b_j(v) c_v(k)$
  Fin Pour
Fin Pour
Pour $i = 1$ **à** N **Faire**
  $\delta_1(i) = \pi_i f_i(o_1)$
Fin Pour
Pour $t = 2$ **à** T **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\delta_t(j) = \max_{1 \leq i \leq N} \{\delta_{t-1}(i) a_{ij}\} f_j(o_t)$
    $\psi_t(j) = \arg \max_{1 \leq i \leq N} \{\delta_{t-1}(i) a_{ij}\}$
  Fin Pour
Fin Pour
$P(V = O, S = Q^*) = \max_{1 \leq i \leq N} \{\delta_T(i)\}$
$q_T^* = \arg \max_{1 \leq i \leq N} \{\delta_T(i)\}$
Pour $t = T - 1$ **à** 1 **Faire**
  $q_t^* = \psi_{t+1}(q_{t+1}^*)$
Fin Pour

on note $X_{1..t}$ les t premiers éléments de la suite X (X pouvant être une suite de variables aléatoires ou de réalisations de variables aléatoires). On définit alors

$$\delta_t(i) = \max_{Q_{1..t-1} \in \mathbb{S}^{t-1}, J_{1..t} \in \mathbb{V}^t} \{P(V_{1..t} = O_{1..t}, I_{1..t} = J_{1..t}, S_{1..t-1} = Q_{1..t-1}, S_t = s_i / \lambda)\}$$

la probabilité du meilleur chemin partiel amenant à l'état caché s_i au temps t lorsque les symboles intermédiaires choisis sont optimaux et $\psi_t(i)$ le meilleur chemin amenant à l'état s_i au temps t à partir du temps $t-1$.

En notant $g_j(k) = \max_{v=1..M} b_j(v) c_v(k)$, on montre aisément les relations suivantes :

$$\begin{aligned}
\delta_1(i) &= \max_{1 \leq v \leq M} \pi b_i(v) c_v(k) \\
&= \pi_i g_i(o_1) \\
\delta_t(j) &= \max_{1 \leq i \leq N, 1 \leq v \leq M} \delta_{t-1}(i) a_{i,j} b_j(v) c_v(k) \\
&= \max_{1 \leq i \leq N} \delta_{t-1}(i) a_{i,j} g_j(k) \\
&= \left(\max_{1 \leq i \leq N} \delta_{t-1}(i) a_{i,j} \right) g_j(k)
\end{aligned}$$

La séquence des symboles intermédiaires V^* peut alors facilement s'obtenir à partir de la séquence des états cachés Q^* et des valeurs $g_j(k)$. L'algorithme « étendu » de Viterbi est alors donné par l'algorithme 5.8. Une version utilisant le ré-échelonnement à l'aide des logarithmes népériens est donnée par l'algorithme 5.9. La complexité des algorithmes est $O(NM^2 + N^2T)$.

5.4 Intérêt des MMCSS pour la classification d'images

Après la conception de ces nouveaux modèles, nous nous sommes posé la question de savoir s'ils apportaient vraiment quelque chose pour une tâche telle que la reconnaissance d'images

Algorithme 5.7: Algorithme de Viterbi MMCSS avec ré-échelonnement

Pour $j = 1$ **à** N **Faire**
  Pour $k = 1$ **à** M **Faire**
    $f_j(k) = \sum_{v=1}^{M} b_j(v) c_v(k)$
  Fin Pour
Fin Pour
Pour $i = 1$ **à** N **Faire**
  $\tilde{\delta}_1(i) = \ln \pi_i + \ln f_i(o_1)$
Fin Pour
Pour $t = 2$ **à** T **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\tilde{\delta}_t(j) = \max_{1 \leq i \leq N} \left\{ \tilde{\delta}_{t-1}(i) + \ln a_{ij} \right\} + \ln f_j(o_t)$
    $\tilde{\psi}_t(j) = \arg \max_{1 \leq i \leq N} \left\{ \tilde{\delta}_{t-1}(i) + \ln a_{ij} \right\}$
  Fin Pour
Fin Pour
$\ln P(V = O, S = Q^*) = \max_{1 \leq i \leq N} \left\{ \tilde{\delta}_T(i) \right\}$
$q_T^* = \arg \max_{1 \leq i \leq N} \left\{ \tilde{\delta}_T(i) \right\}$
Pour $t = T - 1$ **à** 1 **Faire**
  $q_t^* = \tilde{\psi}_{t+1}(q_{t+1}^*)$
Fin Pour

(Aupetit et al., 2004a) (Aupetit et al., 2004b). Comme nous l'avons expliqué précédemment, un système d'intelligence artificielle utilisant des MMC se conçoit en trois phases : le pré-traitement, l'apprentissage et le post-traitement. Ces trois phases peuvent être complexes et dépendantes des données et de la tâche à effectuer. Par conséquent, en prenant un tel système et en changeant uniquement le type de modèle, dans notre cas des MMCSS en lieu et place des MMC, il est fort probable qu'un biais non négligeable soit introduit par une excessive spécialisation du système pour un modèle ou un autre. Ce biais peut provoquer des performances ne traduisant pas forcément les capacités des différents types de modèles. Évaluer l'intérêt d'un type de modèles par un taux de bonnes classifications n'est donc pas un bon indicateur de l'intérêt de chacun.

Les systèmes de classification ou de reconnaissance d'images utilisent en interne la vraisemblance des observations à classer. L'utilisation de cette vraisemblance est très dépendante des données et de la tâche réalisée ; néanmoins elle est classiquement utilisée pour comparer le niveau de « ressemblance » entre les images à classer et les représentants de classes. Nous proposons donc d'utiliser cette vraisemblance, afin d'évaluer la capacité de comparaison des deux types de modèles.

Nous considérons les images de la base ORL (Samaria and Harter, 1994). Cette base est composée de 400 images de dimensions 92×192 pixels en 256 niveaux de gris. Ces images sont réparties en 40 classes, chaque classe correspondant à un individu. Les images présentes dans une classe sont des photographies du visage de la personne sous différents angles. Un problème classique à résoudre avec ce jeu de données est d'affecter les bons visages aux bonnes personnes. Lorsque l'on apprend des images en 256 niveaux de gris à l'aide de MMC, on est habituellement obligé de ré-échantillonner les images avec moins de niveaux de gris (128, 64, ...) afin d'empêcher les MMC d'être trop stricts et de produire une vraisemblance

Algorithme 5.8: Algorithme de Viterbi « étendu » MMCSS

Pour $j = 1$ **à** N **Faire**
  Pour $k = 1$ **à** M **Faire**
    $g_j(k) = \max_{v=1}^{M} b_j(v)c_v(k)$
    $h_j(k) = \arg\max_{v=1}^{M} b_j(v)c_v(k)$
  Fin Pour
Fin Pour
Pour $i = 1$ **à** N **Faire**
  $\delta_1(i) = \pi_i g_i(o_1)$
Fin Pour
Pour $t = 2$ **à** T **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\delta_t(j) = \max\limits_{1\leq i\leq N} \{\delta_{t-1}(i)a_{ij}\}\, g_j(o_t)$
    $\psi_t(j) = \arg \max\limits_{1\leq i\leq N} \{\delta_{t-1}(i)a_{ij}\}$
  Fin Pour
Fin Pour
$P(V = O, I = J^*, S = Q^*/\lambda) = \max\limits_{1\leq i\leq N} \{\delta_T(i)\}$
$q_T^* = \arg \max\limits_{1\leq i\leq N} \{\delta_T(i)\}$
Pour $t = T - 1$ **à** 1 **Faire**
  $q_t^* = \psi_{t+1}(q_{t+1}^*)$
Fin Pour
Pour $t = 1$ **à** T **Faire**
  $j_t^* = h_{q_t^*}(o_t)$
Fin Pour

nulle, car un niveau de gris n'est pas apparu lors de l'apprentissage[1]. Le problème de ce ré-échelonnement des niveaux de gris est qu'il appauvrit les informations contenues dans l'image. Plus ce ré-échantillonnage est important, moins l'image est informative, donc moins les MMC sont discriminants et plus ils risquent de reconnaître tout et n'importe quoi.

Pour mettre en valeur les capacités des MMCSS sans trop pénaliser les MMC, nous avons ré-échantillonné les images en 128 niveaux de gris et nous les avons transformées en séquences par la méthode déjà décrite à la section 4.5.1.1. Les trois premiers visages des cinq premières personnes de la base sont appris plusieurs fois avec l'algorithme GHOSP[2] (cf. section 2.5.4). Le paramétrage de l'algorithme a peu d'importance car on ne cherche pas forcément les modèles optimaux, mais plutôt des modèles de qualité comparable. Le nombre d'états cachés utilisés par les modèles est compris entre 2 et 7.

Pour évaluer la robustesse de l'apprentissage, nous cherchons à savoir pour un visage appris où se situent les 10 visages de la personne dans un classement des 400 images par vraisemblance décroissante. L'objectif est donc que ces 10 visages se trouvent parmi les 10 premières images du classement. Pour cela, nous avons défini quatre indicateurs a, b, $Top10$ et $Top20$ permettant d'évaluer le classement. On note Rank(i) la position de l'image i dans le classement. Les images ayant une vraisemblance nulle sont situées en fin de celui-ci dans un ordre quelconque. Il est donc nécessaire de les traiter séparément. Pour cela, on définit E

[1] La probabilité d'émission d'un niveau de gris n'apparaissant pas à l'apprentissage est généralement nulle lorsque le modèle est optimum local ou global au sens du maximum de vraisemblance. Par exemple, l'algorithme de Baum-Welch fait converger les probabilités d'émission de ces symboles vers 0.

[2] L'algorithme s'adapte sans difficulté aux MMCSS car dans les deux cas les modèles sont définis par les matrices (Π, A, B). Seuls les algorithmes *Forward* et de Baum-Welch ont donc besoin d'être changés.

Algorithme 5.9: Algorithme de Viterbi « étendu » MMCSS avec ré-échelonnement

Pour $j = 1$ **à** N **Faire**
  Pour $k = 1$ **à** M **Faire**
    $g_j(k) = \max_{v=1}^{M} b_j(v)c_v(k)$
    $h_j(k) = \arg\max_{v=1}^{M} b_j(v)c_v(k)$
  Fin Pour
Fin Pour
Pour $i = 1$ **à** N **Faire**
  $\tilde{\delta}_1(i) = \ln \pi_i + \ln g_i(o_1)$
Fin Pour
Pour $t = 2$ **à** T **Faire**
  Pour $j = 1$ **à** N **Faire**
    $\tilde{\delta}_t(j) = \max\limits_{1 \leq i \leq N} \left\{ \tilde{\delta}_{t-1}(i) + \ln a_{ij} \right\} + \ln g_j(o_t)$
    $\tilde{\psi}_t(j) = \arg \max\limits_{1 \leq i \leq N} \left\{ \tilde{\delta}_{t-1}(i) + \ln a_{ij} \right\}$
  Fin Pour
Fin Pour
$\ln P(V = O, I = J^*, S = Q^*/\lambda) = \max\limits_{1 \leq i \leq N} \left\{ \tilde{\delta}_T(i) \right\}$
$q_T^* = \arg \max\limits_{1 \leq i \leq N} \left\{ \tilde{\delta}_T(i) \right\}$
Pour $t = T - 1$ **à** 1 **Faire**
  $q_t^* = \tilde{\psi}_{t+1}(q_{t+1}^*)$
Fin Pour
Pour $t = 1$ **à** T **Faire**
  $j_t^* = h_{q_t^*}(o_t)$
Fin Pour

comme étant l'ensemble des images de la personne à reconnaître dont la vraisemblance est non nulle et L l'ensemble des images dont la vraisemblance est non nulle. On a :

$$a = \begin{cases} 1 & \text{si } |E| = 0 \\ 0 & \text{si } |L| - |E| = 0 \\ \frac{\sum_{i \in E} \text{Rank}(i) - \sum_{k=1}^{|E|} k}{|E|(|L|-|E|)} & \text{sinon} \end{cases}$$

$$b = \frac{10 - |E|}{10}$$

La valeur a indique que les images à reconnaître se situent en haut du classement tandis que b indique la proportion d'images à reconnaître totalement rejetées par le modèle *i.e.* les images à reconnaître de vraisemblance nulle.

Lorsque $|E| = 0$ tous les visages de la personne sont rejetés[3] donc nous imposons $a = 1$. Au contraire, lorsque $|L| - |E| = 0$, seuls les visages de la personne sont reconnus donc ils sont forcément les premiers dans le classement, donc nous imposons $a = 0$. L'objectif est donc que a soit minimum. L'objectif de b est de mesurer le rejet catégorique des images à reconnaître. L'objectif est donc de minimiser b afin de confier les choix irrévocables sur ces images au processus de classification. $Top10$ et $Top20$ indiquent respectivement le nombre de visages à reconnaître situés parmi les 10 et 20 premiers visages du classement dont la vraisemblance est non nulle. L'objectif est donc que leurs valeurs soient maximales.

[3]Ce cas ne doit en principe jamais se produire car le visage appris fait partie de E, sa vraisemblance étant non nulle.

Pour effectuer nos expérimentations, nous avons considéré plusieurs lois de substitution choisies et paramétrées arbitrairement. Les lois, que nous avons utilisées, sont définies par la formule générale :

$$c_v(k) = \frac{f(v,k)}{\sum_{l=1}^{M} f(v,l)}$$

Les fonctions f utilisées sont :

- $f_0(v,k) = \begin{cases} 1 & \text{si } v = k \\ 0 & \text{sinon} \end{cases}$. Cette fonction correspond au cas de l'utilisation de MMC ordinaire.
- $f_1(v,k) = \max(-|v-k| + 12.5, 0)$
- $f_2(v,k) = \max(-|v-k| + 12.5, 0.01)$
- $f_3(v,k) = \max(-\ln \frac{|v-k|}{4}, 0)$
- $f_4(v,k) = \max(-\ln \frac{|v-k|}{16}, 0)$
- $f_5(v,k) = \max(-\ln \frac{|v-k|}{16}, 0.01)$
- $f_6(v,k) = \begin{cases} 1 & \text{si } k \in [v-1; v+1] \\ 0 & \text{sinon} \end{cases}$
- $f_7(v,k) = \begin{cases} \sqrt{25-(v-k)^2} & \text{si } 25-(v-k)^2 \geq 0 \\ 0 & \text{sinon} \end{cases}$
- $f_8(v,k) = \begin{cases} 1 & \text{si } v = k \\ \frac{1}{|v-k|} & \text{sinon} \end{cases}$
- $f_9(v,k) = e_k$ avec e_k le nombre de fois qu'apparaît le niveau de gris k dans l'image apprise.

Ces lois établissent toutes une certaine « équivalence » des niveaux de gris « proches », comme pourrait le faire un système plus complexe. Nous avons considéré ces lois car elles définissent différentes formes dans la quantification des « équivalences » (cf. figure 5.2). La variation des probabilités dans les lois sont linéaires, logarithmiques, en créneaux, quadratiques, proportionnelles à la fonction inverse ou statistiques.

Le tableau 5.1 présente les résultats de ces expérimentations. Les valeurs présentées sont des moyennes obtenues sur les trois premiers visages des cinq premières personnes pour plusieurs apprentissages.

	a	$\sigma(a)$	b	$\sigma(b)$	$Top10$	$Top20$
f_0	0.000	0.000	0.900	0.000	1.000	1.000
f_1	0.177	0.219	0.195	0.259	3.667	4.867
f_2	0.128	0.150	0.000	0.000	4.083	5.500
f_3	0.174	0.214	0.231	0.270	3.683	4.817
f_4	0.156	0.193	0.261	0.259	3.800	4.783
f_5	0.129	0.153	0.000	0.000	4.450	5.650
f_6	0.181	0.239	0.731	0.193	1.433	1.783
f_7	0.184	0.226	0.168	0.228	3.800	5.133
f_8	0.158	0.187	0.000	0.000	3.750	5.650
f_9	0.079	0.144	0.860	0.080	1.067	1.200

TAB. 5.1 – Valeurs moyennes des indicateurs sur les classements. $\sigma(x)$ représente l'écart-type moyen de la variable x.

Nous pouvons remarquer que l'apport de MMCSS à ce type d'apprentissage est significatif, surtout au niveau du rejet catégorique des visages appartenant à la classe qui se trouve

fortement amoindri, voire annulé dans le cas où n'importe quel symbole peut être substitué avec n'importe quel autre, mais avec une faible probabilité. Cette propriété permet d'utiliser la vraisemblance comme mesure non binaire de similarité. On remarque également que les MMCSS reconnaissent plus de visages de la classe, mais ceux-ci ne sont pas forcément ceux situés en premier dans le classement. Cependant, on remarque que, pour certaines substitutions, plus de 40% des visages de la classe sont parmi les 10 premiers du classement. Il faut également noter que l'apprentissage des MMC ou des MMCSS a été effectué avec relativement peu d'états cachés, ce qui explique qu'à peine plus de 50% des visages à reconnaître sont parmi les 20 premières images. Des tests réalisés avec plus d'états permettraient certainement d'améliorer ce pourcentage. Quoi qu'il en soit, nous avons montré que l'ajout de connaissances sous la forme d'une loi de substitution bien choisie permet d'améliorer significativement l'apprentissage de séquences. D'une certaine façon, l'ajout d'une loi de substitution donne la responsabilité au modèle de Markov d'effectuer le « ré-échantillonnage » des symboles, de façon à modéliser au mieux la séquence.

5.5 Intérêt des MMCSS pour la segmentation d'images

Afin d'évaluer les capacités des MMCSS pour la segmentation d'images, nous avons appris et segmenté les images de la figure 5.3 après les avoir ré-échantillonnées sur 128 niveaux de gris et les avoir linéarisées en séquences comme précédemment. Les séquences ont été apprises par l'algorithme GHOSP (cf. section 2.5.4) avec un paramétrage classique, sans importance pour la suite, et un nombre d'états cachés compris entre 10 et 15.

Pour effectuer nos expérimentations, nous avons considéré quatre lois de substitution choisies et paramétrées arbitrairement. Les lois que nous avons utilisées sont définies par la formule générale :

$$c_v(k) = \frac{f(v,k)}{\sum_{l=1}^{M} f(v,l)}$$

Les fonctions f utilisées sont :

- $f_0(v,k) = \begin{cases} 1 & \text{si } v = k \\ 0 & \text{sinon} \end{cases}$. Cette fonction correspond au cas de l'utilisation de MMC ordinaire.
- $f_1(v,k) = \max(-|v-k| + 12.5, 0)$
- $f_2(v,k) = \max(-|v-k| + 12.5, 0.01)$
- $f_3(v,k) = \frac{1}{12.5\sqrt{2\pi}} e^{-\frac{(v-k)^2}{12.5^2}}$

Cette liste de fonction a été choisie car elle permet de comparer des formes très différentes de lois de substitution : une forme linéaire, une forme linéaire mais non nulle et une forme gaussienne.

Après segmentation par l'algorithme de Viterbi (MMC ou MMCSS) ou Viterbi « étendu », les états correspondant aux éléments de la séquence sont colorés de manière à ce que tous les pixels correspondant au même état aient la même couleur. Les images ainsi reconstituées sont données par les figures 5.4, 5.5, 5.6, 5.7, 5.8 et 5.9.

Plusieurs caractéristiques émergent de ces expérimentations :

- La séquence d'états cachés obtenue à partir de l'algorithme de Viterbi avec des MMCSS est composée d'un nombre plus réduit de couleurs (donc d'états) avec des zones plus grandes et homogènes par rapport aux MMC. Les MMCSS ont donc la capacité d'effectuer des segmentations plus homogènes et donc potentiellement plus utiles dans certains cas.
- L'utilisation d'une loi de substitution dont les probabilités sont strictement positives rend la segmentation moins discriminante. Cette réduction de la discrimination est par-

ticulièrement visible sur les séquences de symboles intermédiaires obtenus par l'algorithme de Viterbi « étendu ». Lorsque l'image est très perturbée, comme dans le cas des images satellites, on obtient des séquences de symboles intermédiaires et/ou d'états quasiment uniformes.
- Les séquences d'états obtenues par Viterbi et f_0 (*i.e.* par des MMC) et celles obtenues avec Viterbi « étendu » sont très dépendantes de la méthode de linéarisation de l'image. En effet, on distingue très clairement les blocs de pixels. Au contraire, dans le cas de Viterbi avec f_1, f_2 ou f_3, la segmentation obtenue ne semble pas être très influencée par ce découpage.
- Finalement, on remarque que l'algorithme de Viterbi « étendu » produit des séquences d'états plus homogènes que l'algorithme de Viterbi, et des séquences de symboles intermédiaires qui réalisent une sorte d'échantillonnage automatique et adapté de l'image.

Par conséquent, et à condition de bien choisir la loi de substitution, la segmentation d'images à l'aide de MMCSS offre un intérêt certain par rapport aux MMC.

5.6 Conclusion

Nous avons défini un nouveau type de modèle de Markov caché ayant la capacité d'incorporer des connaissances *a priori* sur le problème, afin d'améliorer son efficacité. Nous avons montré que les algorithmes standards des MMC pouvaient être, sans trop de difficulté, adaptés aux MMCSS. Nous avons également montré que, hormis un pré-calcul, la complexité de ces algorithmes est identique, alors que les capacités des MMCSS sont supérieures à celles des MMC. Finalement, nous avons montré que les MMCSS possèdent de bonnes propriétés promettant des avancées fructueuses, notamment en traitement d'image.

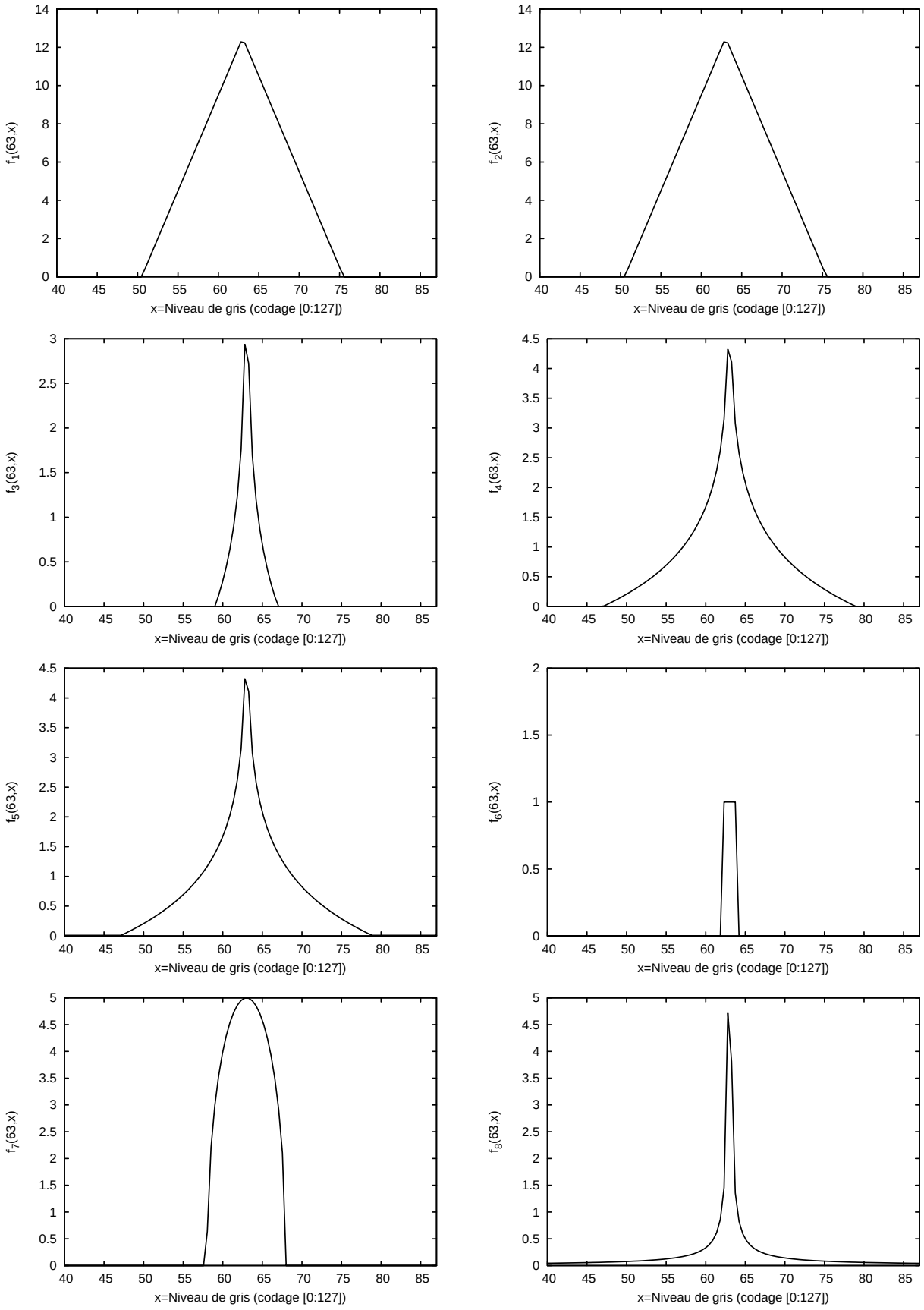


FIG. 5.2 – Formes des fonctions utilisées pour la construction des lois de substitution.

FIG. 5.3 – Images utilisées pour les expérimentations.

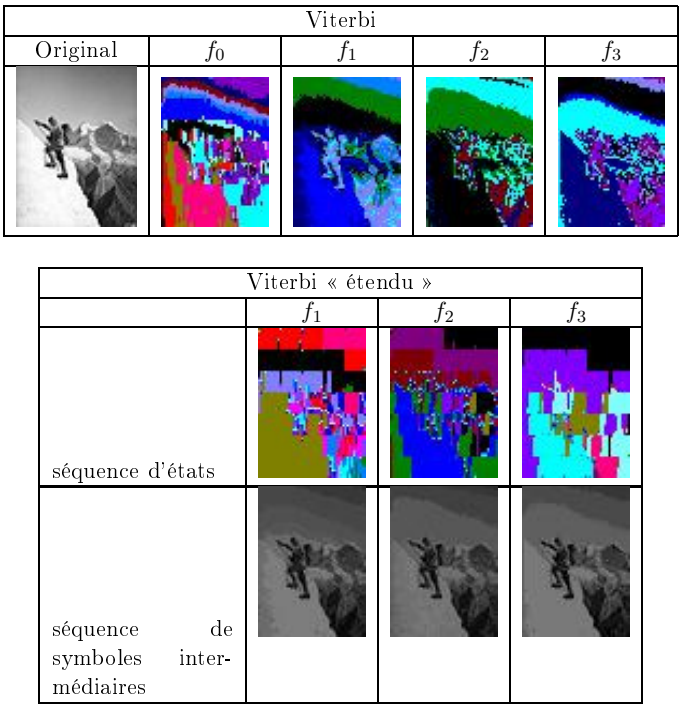
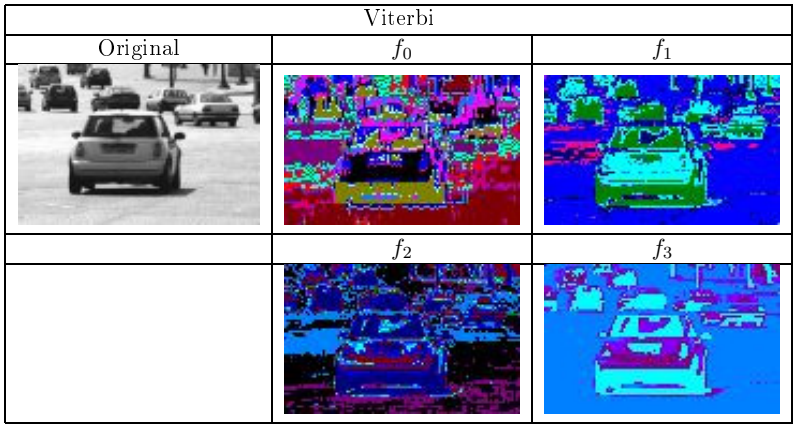


FIG. 5.4 – Résultats de la segmentation de l'image (a) de la figure 5.3.



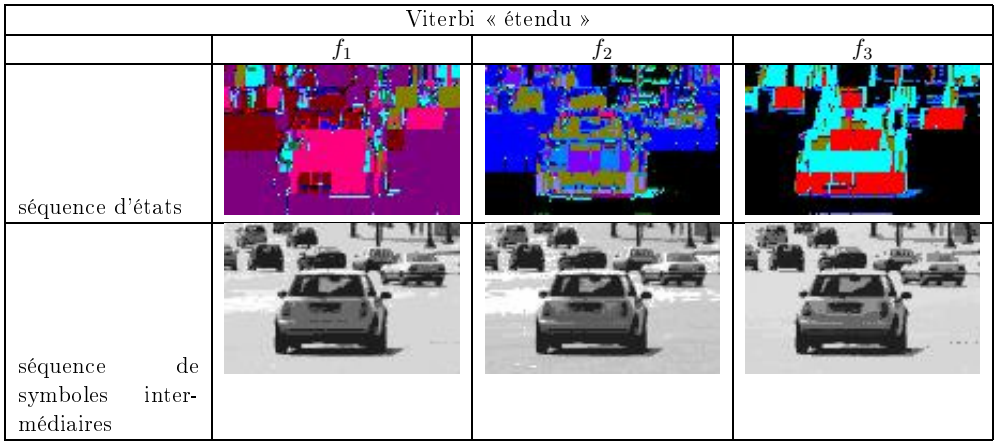
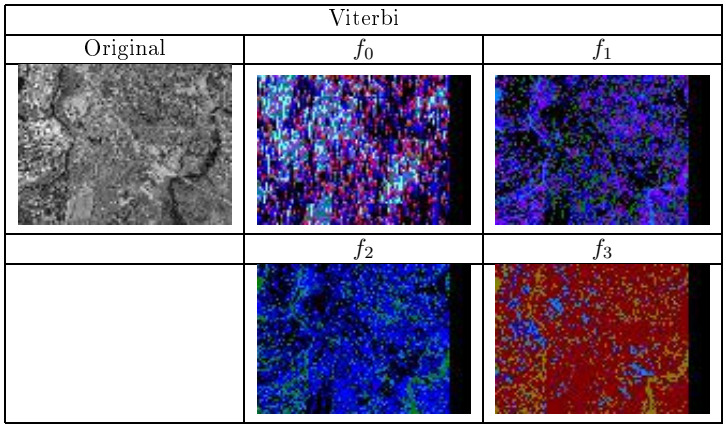


FIG. 5.5 – Résultats de la segmentation de l'image (b) de la figure 5.3.



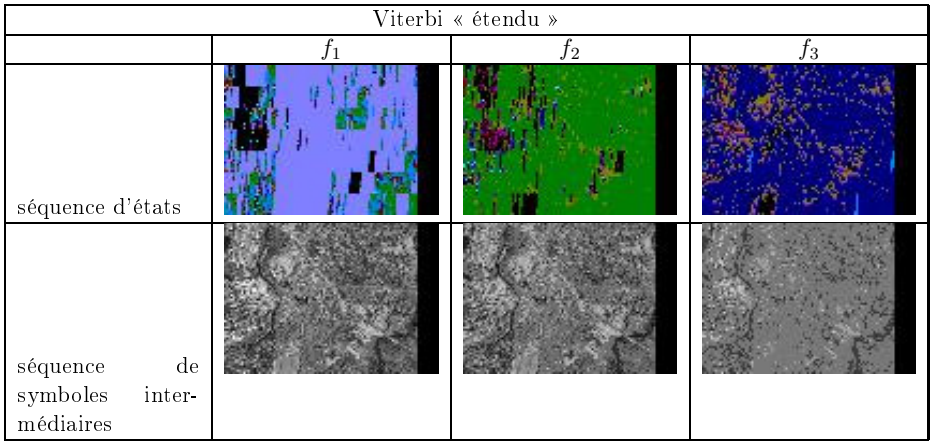


FIG. 5.6 – Résultats de la segmentation de l'image (c) de la figure 5.3.

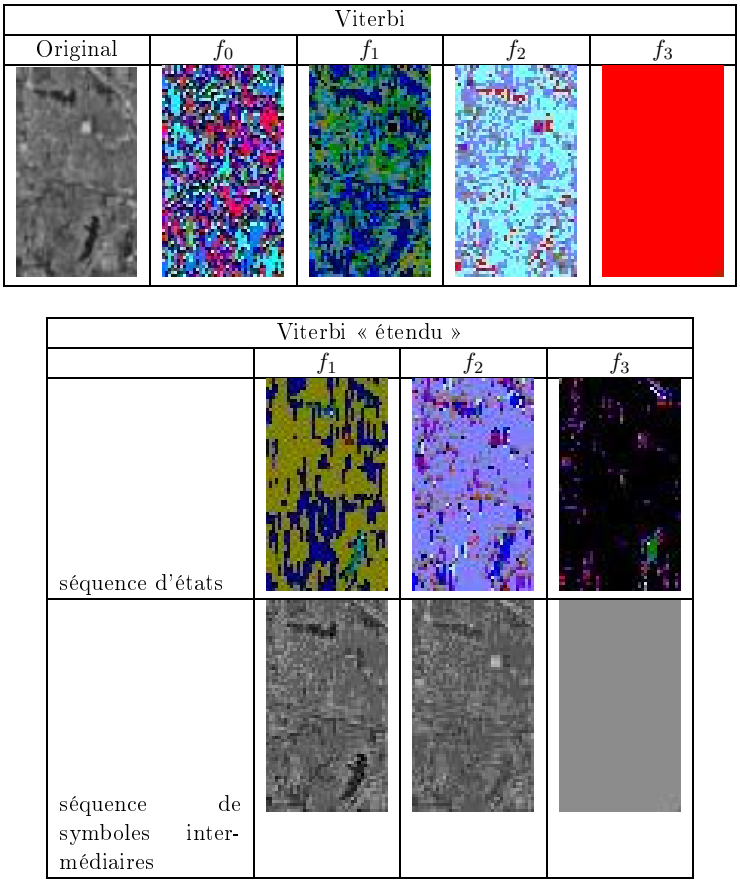


FIG. 5.7 – Résultats de la segmentation de l'image (d) de la figure 5.3.

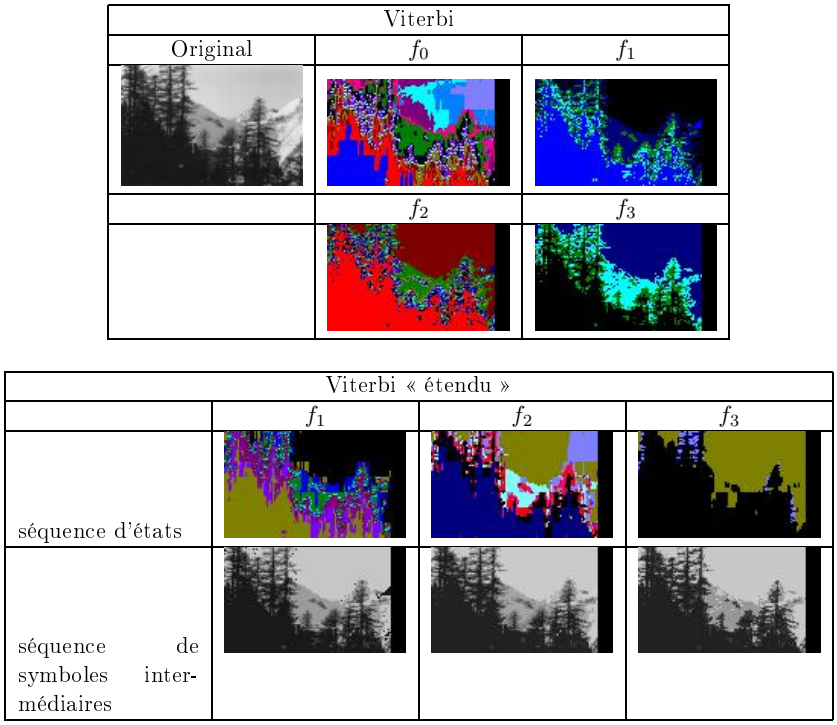
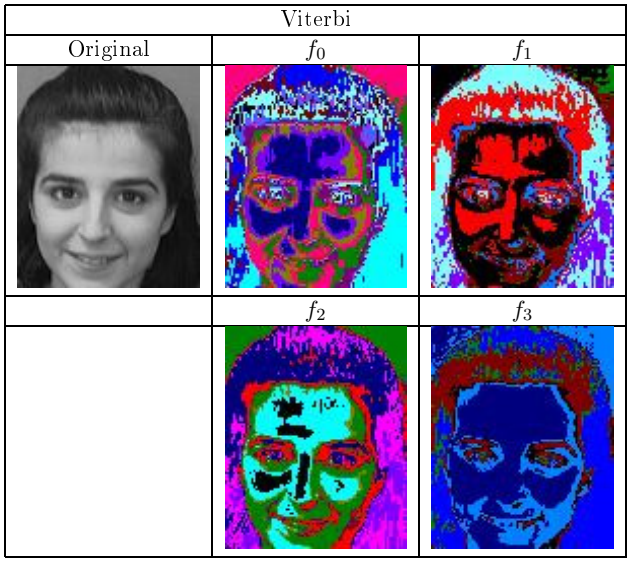


FIG. 5.8 – Résultats de la segmentation de l'image (e) de la figure 5.3.



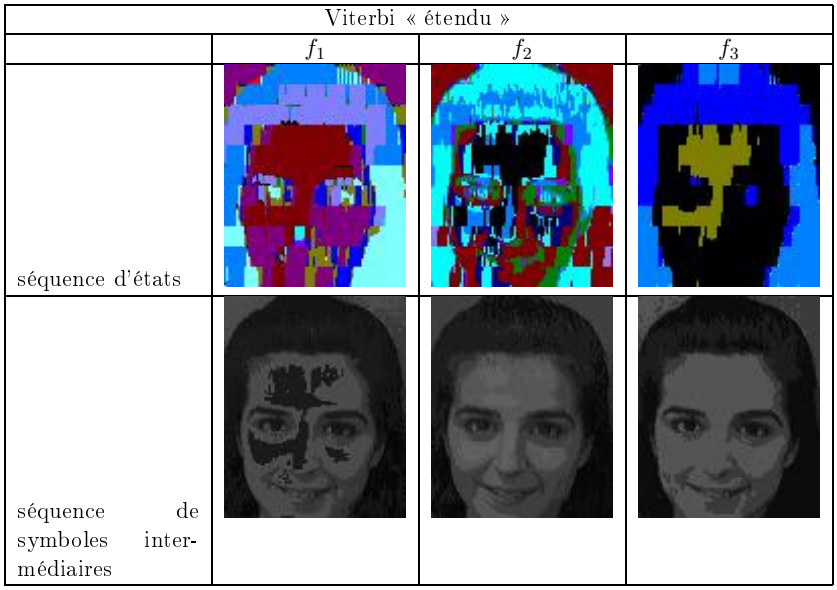


FIG. 5.9 – Résultats de la segmentation de l'image (f) de la figure 5.3.

Chapitre 6

Matrice de *scatterplots* pseudo-euclidienne

6.1 Introduction

Les systèmes d'intelligence artificielle (SIA) utilisant des MMC sont devenus au fil du temps de plus en plus complexes au point qu'effectuer une modification dans le système peut conduire à des gains ou à des pertes significatives de performances, sans que l'on sache forcément ce qui en est la cause. Parallèlement, depuis quelques années, la visualisation de données est de plus en plus utilisée pour l'analyse de données collectées expérimentalement. Afin d'améliorer les connaissances que l'on peut avoir des SIA utilisant des MMC, nous nous sommes tournés vers la visualisation de données. Pour cela, nous aurions pu suivre plusieurs pistes : la visualisation des modèles et de leurs structures, la visualisation du processus d'apprentissage des MMC, la visualisation de l'effet des MMC sur des données telles que la segmentation, ... La piste que nous avons choisie de prendre consiste à essayer de comprendre comment un ensemble de MMC obtenues à partir de séquences d'observations s'organise dans l'espace.

Les travaux décrits dans ce chapitre ont donné lieu à l'encadrement d'un projet de fin d'études (Thomy, 2003).

6.1.1 L'approche multi-dimensionnelle

Le premier moyen que nous avons envisagé consiste à utiliser les coefficients des matrices comme étant les coordonnées des MMC dans l'espace $\Lambda \subset \mathbb{R}^{N(N+M+1)}$. Cette approche n'est cependant pas été très intéressante pour plusieurs raisons :

- le nombre de dimensions est trop important : par exemple, si $N = 4$ et $M = 6$ alors les MMC se situent dans un espace de 44 dimensions. On peut également noter que les MMC ont typiquement beaucoup plus d'états cachés et de symboles que cet exemple. Ce nombre trop important de dimensions écarte donc la majorité des techniques de visualisation. En effet, ces dernières sont exploitables avec relativement peu de dimensions, mais avec beaucoup de données. Dans notre cas, il y a beaucoup de dimensions mais relativement peu de données.
- Λ est un espace convexe mais sa forme n'est pas habituelle en raison des contraintes de stochasticité des matrices (A, B, Π). Ces contraintes sont donc très susceptibles de faire apparaître des motifs prédominants mais néanmoins non intéressants. Considérer $(\Lambda^*, \boxplus, \boxdot)$ pourrait réduire ces effets, mais le problème du grand nombre de dimensions se poserait toujours.

– Un MMC obtenu par le critère du maximum de vraisemblance est au moins aussi bon qu'un autre MMC obtenu à partir du premier, en renumérotant les états cachés. Ces MMC ont potentiellement des coordonnées très différentes dans l'espace Λ. Donc, avec une technique classique de visualisation de données multidimensionnelle, l'utilisateur aboutira à la conclusion que ces MMC sont très différents, alors que ce n'est pas le cas.

Trouver un paramétrage (une représentation multidimensionnelle) des MMC capable de répondre à tous ces problèmes est très difficile. L'approche multi-dimensionnelle étant peu réalisable, nous nous sommes tournés vers une autre approche utilisant la notion de dissimilarité.

6.1.2 L'approche par dissimilarité

Le deuxième moyen que nous avons envisagé consiste à utiliser une dissimilarité, afin de définir la « ressemblance » de deux MMC. Cette approche possède un avantage majeur : sa flexibilité. En effet, on peut utiliser un algorithme aussi complexe que nécessaire pour définir ses valeurs. On peut mesurer ce que l'on veut. Par exemple, intégrer le fait que les états puissent être renumérotés peut s'effectuer sans trop de difficulté. Si ces MMC sont utilisés pour calculer une mesure au sein d'un algorithme de classification, il devient possible de comprendre comment cette mesure perçoit et organise les objets dans l'espace, ... Cette approche semble très intéressante, mais elle possède un inconvénient majeur : sa flexibilité. En effet, sa flexibilité offre une grande liberté pour définir ses valeurs mais, en conséquence, elle vérifie peu de propriétés mathématiques, ce qui rend difficile l'analyse des configurations. Les seules propriétés mathématiques imposées à une dissimilarité en analyse de données w sont :

– la symétrie : pour tout objet x et y, on a $w(x, y) = w(y, x)$
– la positivité[1] : pour tout objet x et y, on a $w(x, y) \geq 0$
– l'auto-similarité : pour tout objet x, on a $w(x, x) = 0$

Ces propriétés se prêtent bien à la recherche de groupes dans les données par des techniques telles que les *Shaded Similarity Matrix* ou la visualisation de graphe, mais pouvoir appréhender des notions telles que l'organisation des objets les uns par rapport aux autres ou la notion d'échelle dans les valeurs de la dissimilarité, reste difficile.

Une solution partielle à la notion d'organisation des objets peut être fournie par les techniques de *Multi Dimensional Scaling* (MDS). Le MDS consiste à projeter les objets dans un espace de faible dimension, typiquement 2 ou 3, en minimisant un critère particulier. Les critères utilisables peuvent être très variés (minimisation de l'écart entre la distance projetée et la dissimilarité, minimisation des inversions d'ordre dans les objets[2], ...) (Cox and Cox, 2001) (Borg and Groenen, 1997). Les MDS permettent donc de voir l'organisation des objets selon une certaine projection. Que le MDS soit obtenu par un MDS ordinal ou par la minimisation d'un critère numérique, il est possible que la projection obtenue soit peu représentative des échelles dans les valeurs de dissimilarité (ex : deux groupes d'objets apparaissent, mais semblent également partiellement se recouvrir, alors que du point de vue de la dissimilarité ces deux groupes sont parfaitement bien séparés.) Ces informations d'échelles ont relativement peu d'importance lorsqu'on analyse des données multidimensionnelles ou des dissimilarités de manière classique. Mais, lorsque l'on cherche à comprendre ce que modélise un ensemble d'objets tels que des MMC, on cherche plus à analyser les répartitions spatiales et les relations entre les objets qu'à détecter ou confirmer des groupes. Par conséquent, il est nécessaire de pouvoir représenter ces relations à l'aide de techniques particulières.

[1] Dans la suite, nous montrerons que cette propriété n'est pas nécessaire dans les visualisations que nous proposons.

[2] On parle de MDS ordinal : si $w(x, y) < w(x, z)$ alors les points images obtenus par la projection vérifient le même type de relation pour la norme usuelle.

6.2 Les limites du principe de « similarité/proximité »

La majorité des techniques de visualisation, qu'elles soient pour des données multidimensionnelles ou pour des dissimilarités, utilisent de manière plus ou moins explicite un principe que nous nommerons dans la suite le principe de « similarité/proximité » :

deux objets x et y sont similaires si leurs images X et Y sont « proches ».

On peut remarquer que les techniques de visualisation de données appliquent de manière générale un processus en deux phases (cf. figure 6.1) pour construire les visualisations. La première phase consiste à trouver des représentants des objets. La deuxième phase consiste alors à projeter ces représentants dans un espace de dimension plus faible afin de visualiser les objets. Ces deux phases sont étroitement liées, mais jouent des rôles différents : la phase de représentation travaille sur les objets presque indépendamment de la visualisation, tandis que la deuxième phase effectue une « projection » des données afin de permettre leurs analyses. Par exemple, dans le cas de données multidimensionnelles et de la technique de la matrice de *scatterplots* ou de la technique RadViz, la première phase correspond à l'identité ou à la sélection de certaines dimensions du système de coordonnées, tandis que la deuxième phase consiste à projeter les points images sur les différents *scatterplots* ou selon la technique RadViz. Dans le cas d'une dissimilarité et d'un MDS visualisé par une analyse en composantes principales, la phase de représentation pourrait consister à déterminer des coordonnées pour les points dans un espace réel de dimension K, tandis que la phase de projection consiste à ne représenter ces points que sur deux axes principaux. Dans les deux phases, le principe de « similarité/proximité » est généralement respecté afin qu'il le soit globalement.

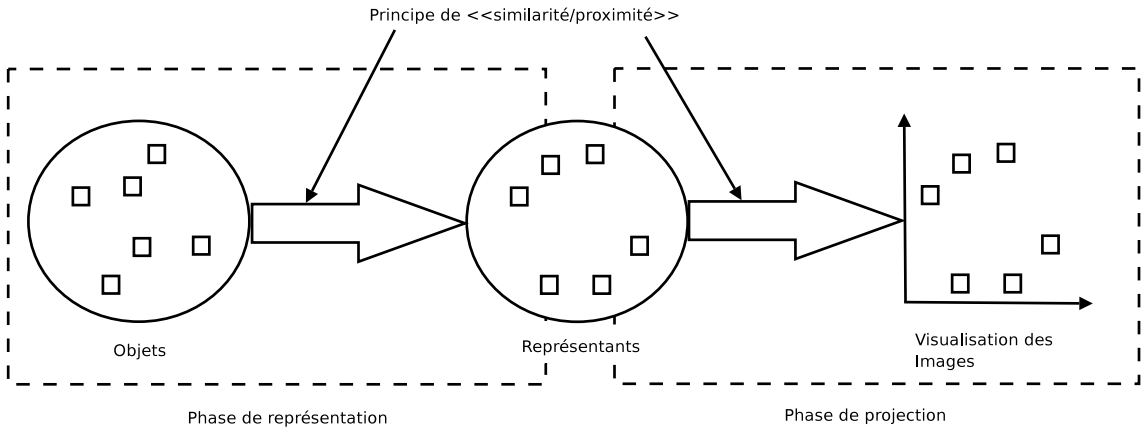


FIG. 6.1 – Structure en 2 phases de la transformation des données dans une visualisation.

Ce principe, bien que valable dans de nombreux cas, peut être trop strict dans certains autres. Prenons l'exemple d'une dissimilarité établie par un procédé quelconque. Supposons l'existence de trois objets x, y et z dont la phase de représentation a abouti aux points X, Y et Z dans un espace vectoriel quelconque. Supposons, sans perte de généralité, que la phase de projection est l'identité.

Si l'on est capable de déterminer que la dissimilarité est une distance euclidienne grâce à l'une des propriétés données dans (Gower and Legendre, 1986) alors le principe de « similarité/proximité » est vérifié. Dans le cas où l'on ne peut pas le décider ou dans le cas d'une dissimilarité non euclidienne, on peut être confronté à une dissimilarité ne vérifiant pas le principe de « similarité/proximité ».

Considérons un exemple. Si x est similaire à y et que x est similaire à z alors il est naturel que y et z soient similaires[3]. Le principe de « similarité/proximité » impose alors que les vecteurs X, Y et Z soient relativement proches dans l'espace : leurs coordonnées dans l'espace vectoriel sont proches. Mais, en fonction de la dissimilarité considérée, il est tout à fait possible que y et z soient très différents. Selon le principe de « similarité/proximité », les vecteurs X, Y et Z sont proches. Mais comme y est très différent de z, on devrait pouvoir avoir Y et Z très différents. Le principe de « similarité/proximité » a donc un effet de contamination de proche en proche. Cet effet est souhaitable dans de nombreux cas mais, lorsque l'on manipule une dissimilarité obtenue par un algorithme, cette contamination peut provoquer des déformations très importantes dans la visualisation, la rendant inutile.

Deux approches sont généralement choisies dans de tels cas : soit la déformation est tolérable, soit elle ne l'est pas. Dans le cas où elle est tolérable, la dissimilarité est approchée[4] par une dissimilarité vérifiant le principe de « similarité/proximité ». Dans le cas où la déformation risque d'être trop importante, il est nécessaire de se passer du principe de « similarité/proximité » et de fournir des outils adaptées à l'analyse. Nous avons choisi de suivre cette deuxième voie.

6.3 Les origines de la méthode proposée

La méthode que nous proposons trouve ses origines dans plusieurs travaux :
- le *Multi Dimensional Scaling* (MDS) classique (Torgeson, 1965) avec notamment les formules de Torgeson pour le centrage du nuage de points, le passage de la norme au produit scalaire et le calcul des représentants dans un espace euclidien
- les méthodes à noyaux (Vapnik, 1995) utilisées en apprentissage avec notamment l'analyse en composante principale à noyaux (Schölkopf et al., 1999)
- les travaux récents sur l'utilisation de noyaux non définis pour l'apprentissage et la classification (Pekalska et al., 2002)

Le MDS classique (Torgeson, 1965) s'attache à trouver des représentants des objets dans un espace de faible dimension de manière à ce que la distance euclidienne entre deux de ces objets soit aussi proche que possible de la valeur de la dissimilarité entre ces deux objets.

Le principe général au coeur des méthodes à noyaux (Vapnik, 1995), telles que les machines à vecteurs supports (Schölkopf et al., 1999), consiste à projeter les points dans un espace de Hilbert[5] H de plus grande dimension, dans lequel le problème est plus simple à résoudre. Afin de ne pas faire croître la difficulté du problème avec la dimension de l'espace H, on ne calcule pas explicitement la projection des points dans H. En lieu et place, on utilise le produit scalaire pour manipuler ces points. La projection dans un espace H s'effectue en définissant les valeurs du produit scalaire entre les points dans H. La définition des valeurs du produit scalaire peut être réalisée soit en donnant les valeurs prises par ce produit scalaire entre les points, soit en définissant l'expression analytique de ce produit scalaire. Pour que $(x|y)$ soit un produit scalaire, d'après le théorème de Mercer (Guéret et al., 2003), il faut que :
- dans le cas d'un produit scalaire sous forme numérique : la matrice G des valeurs du produit scalaire pour tous les couples de points soit symétrique semi définie positive ;
- dans le cas d'un produit scalaire sous forme analytique : $(x|y)$ soit symétrique et pour toutes fonctions réelles g non nulles telles que $\int g(x)^2 \partial x < \infty$, on ait :

$$\int (x|y)g(x)g(y)\partial x \partial y > 0$$

[3] y et z peuvent être similaires à un degré moindre que les couples (x, y) et (x, z) mais ils restent similaires.

[4] L'approximation de la dissimilarité peut prendre de nombreuses formes : pour préserver l'ordre relatif des objets les uns par rapport aux autres, si D est la matrice des dissimilarités, alors e^D peut être utilisé, ...

[5] Les espaces de Hilbert sont une généralisation des espaces euclidiens en dimension quelconque. Un espace euclidien est un espace vectoriel de dimension finie, muni d'un produit scalaire.

Ces conditions sont très contraignantes cependant quelques travaux récents ont permis de réduire ces contraintes (Boughorbel et al., 2005) sans les dépasser totalement.

Dans la suite de ce chapitre, nous nous plaçons dans le cas d'espaces de dimension finie. Par conséquent, nous emploierons l'expression « espace euclidien » en lieu et place de l'expression « espace de Hilbert ».

Depuis peu, une nouvelle voie est explorée : l'utilisation de noyaux ne vérifiant pas les conditions du théorème de Mercer. Dans cette optique, on ne considère plus H comme un espace de Hilbert, mais plutôt comme un espace pseudo-euclidien *i.e.* muni d'un produit scalaire pseudo-euclidien. Dans (Haasdonk, 2003), (Pekalska et al., 2002), (Ong et al., 2004) et (Graepel et al., 1999), il a été montré que la contrainte des espace de Hilbert sur le produit scalaire n'était pas nécessaire.

Nous avons cherché à combiner ces trois méthodes afin de construire une analyse en composantes principales sur un espace pseudo-euclidien permettant l'analyse visuelle de dissimilarité.

6.4 Espaces pseudo-euclidiens et analyse en composantes principales à noyau indéfini (ACPNI)

Afin de présenter notre méthode, nous allons procéder en plusieurs étapes. Dans un premier temps, nous établissons plusieurs définitions et notations utiles pour la suite. Nous montrons que tout ensemble d'objets, organisé selon une dissimilarité, peut être plongé dans un espace quadratique. Pour faciliter la compréhension de la méthode, nous montrons que le plongement dans un espace euclidien de dimension minimale permet de redémontrer les formules de la technique d'analyse en composantes principales à noyau (ACPN) (Schölkopf et al., 1999). L'application de la même démarche au plongement dans un espace pseudo-euclidien de dimension minimale nous permet d'introduire ce que nommons l'analyse en composantes principales à noyau indéfini (ACPNI). Finalement, nous montrons comment centrer le nuage de points et nous montrons l'unicité, à un isomorphisme près, du plongement dans le cas pseudo-euclidien.

6.4.1 Définitions et notations

Soit $\mathcal{G}$ l'ensemble des objets considérés. Ces objets peuvent être aussi variés que des sons, des luminosités, des images, les caractéristiques des images, des séries temporelles, des modèles de Markov cachés ou des points provenant d'un espace vectoriel multidimensionnel.

Définition 14 *Une fonction $w : \mathcal{G} \times \mathcal{G} \mapsto \mathbb{R}$ est une dissimilarité si et seulement si :*

$$\forall\ x \in \mathcal{G},\ w(x,x) = 0$$
$$\forall\ (x,y) \in \mathcal{G}^2,\ w(x,y) = w(y,x)$$
$$\forall\ (x,y) \in \mathcal{G}^2,\ w(x,y) \geq 0$$

Nous montrerons, dans la suite de ce chapitre, que la dernière caractéristique d'une dissimilarité n'est pas nécessaire pour notre méthode.

Définition 15 *Une dissimilarité w est régulière si et seulement si $\forall\ (x,y) \in \mathcal{G}^2$, on a*

$$w(x,y) = 0 \implies x = y$$

On peut dès lors remarquer que, dans de nombreuses applications, $\mathcal{G}$ est un sous-ensemble d'un espace euclidien H, et que la fonction $w(\cdot,\cdot)$ est définie par $w(x,y) = ||x-y||_H^2$ avec $||\cdot||_H$ la norme associée au produit scalaire $(\cdot|\cdot)_H$ sur l'espace H.

La définition que nous donnons d'une dissimilarité est très générale. Elle inclut la possibilité de considérer des objets n'ayant pas de représentants dans un espace euclidien en utilisant la technique classique du *Multi Dimensional Scaling* introduite par Torgeson (Torgeson, 1965). Lorsque le MDS classique n'est pas utilisable pour obtenir une représentation pertinente, plusieurs alternatives peuvent être envisagées (Borg and Groenen, 1997), telles que le MDS non métrique, mais, dans la grande majorité des cas, le plongement des objets dans l'espace multidimensionel est non linéaire, ce qui rend l'interprétation difficile. Un autre stratégie utilisable consiste à construire une matrice de corrélation G à partir de la dissimilarité et à calculer une analyse en composantes principales (ACP) approchée en ne considérant que les valeurs propres positives lors de la décomposition spectrale. Cependant, lorsque la somme des valeurs propres négatives G est relativement grande en magnitude par rapport à la somme des valeurs propres positives, l'approximation de la projection des points sur les axes principaux peut être très mauvaise et conduire à une mauvaise interprétation des données. Cependant, lorsque la somme des valeurs propres négatives G est relativement négligeable, il a été montré que l'approximation obtenue est de bonne qualité et que l'écart réel/approximation est porteur d'information ou de bruit (Camiz, 2002). Le dernier type de solutions envisageable consiste à approcher la matrice G par une matrice $\tilde{G}$, de manière à ce que $\tilde{G}$ soit positive. (Schnabel and Eskow, 1999), (Cheng and Higham, 1998) et (Cox and Cox, 2001) détaillent plusieurs façons d'effectuer cette approximation.

La méthode que nous proposons utilise une approche similaire au MDS classique, si ce n'est que le plongement s'effectue dans un espace qui n'est pas euclidien. Pour réaliser ce plongement, il est nécessaire d'introduire quelques définitions et notations qui nous seront utiles pour justifier la démarche.

Soit $\mathcal{E}$ un espace vectoriel réel de dimension finie s. Une forme quadratique q sur $\mathcal{E}$ est définie à partir d'une forme symétrique bilinéaire $< \cdot|\cdot >: \mathcal{E}\times\mathcal{E} \rightarrow \mathbb{R}$ par la relation $q(x) =< x|x >$. Le couple $(\mathcal{E}, q)$ est appelé un espace quadratique. De plus, la relation de polarisation permet d'écrire :

$$< x|y >= \frac{q(x+y) - q(x) - q(y)}{2}$$

Soit $Ker(q) = \{x \in \mathcal{E}/\forall\ y \in \mathcal{E},\ < x|y >= 0\}$ le noyau de q. La forme quadratique q (ou sa forme polaire $< \cdot|\cdot >$) est dite singulière si et seulement si le noyau $Ker(q)$ est réduit au singleton $\{0\}$ *i.e.* $Ker(q) = \{0\}$. L'espace $(\mathcal{E}, q)$ est dit pseudo-euclidien si q est non singulière. L'espace $(\mathcal{E}, q)$ est dit euclidien si q est définie positive *i.e.* $q(x) > 0$ pour tout $x \neq 0$.

A chaque espace quadratique $(\mathcal{E}, q)$ peut être associé l'index de Sylvester également appelé signature. Cet index est une caractéristique linéairement invariante de l'espace. Il se présente sous la forme d'un couple d'entiers positifs $\sigma(q) = (p, n)$. Chaque espace quadratique $(\mathcal{E}, q)$ peut être décomposé sous la forme d'une somme directe : $\mathcal{E} = \mathcal{E}^+ \oplus \mathcal{E}^- \oplus \mathcal{E}^0$. Cette décomposition peut être choisie orthogonale ou non.

Soit q_+, q_- et q_0, les restrictions de q sur $\mathcal{E}^+$, $\mathcal{E}^-$ et $\mathcal{E}^0$. q_+ et q_- sont définies positives et q_0 est triviale de valeur nulle. Tout élément $x \in \mathcal{E}$ possède une décomposition unique $x = x^+ + x^- + x^0$ avec $x^+ \in \mathcal{E}^+$, $x^- \in \mathcal{E}^-$ et $x^0 \in \mathcal{E}^0$. On a $q(x) = q_+(x^+) - q_-(x^-)$. Soit $||\cdot||_+$ et $||\cdot||_-$ les normes euclidiennes associées aux espaces $(\mathcal{E}^+, q_+)$ et $(\mathcal{E}^-, q_-)$. Soit $(\cdot|\cdot)_+$ et $(\cdot|\cdot)_-$ les produits scalaires associés à ces normes. On a, pour tout $(x, y) \in \mathcal{E}^2$:

$$q(x) =< x|x >= ||x^+||_+^2 - ||x^-||_-^2$$

$$< x|y >= (x^+|y^+)_+ - (x^-|y^-)_-$$

Dans l'étude des espaces pseudo-euclidiens, le cône isotropique défini par l'équation $C(q) = \{x \in \mathcal{E}/q(x) = 0\}$ joue un rôle très important. Ce cône correspond à l'ensemble des points qui vérifient l'équation $||x + ||_+^2 = ||x^-||_-^2$.

Dans la suite, on suppose que $\mathcal{E} = \mathbb{R}^s$ ($s \in \mathbb{N}^*$). Par convention, on note A' la matrice transposée de A. On suppose, sauf indication contraire, que tout espace $\mathbb{R}^s$ est muni du produit scalaire habituel $(X|Y) = Y'X = X'Y$. La norme euclidienne associée sera notée $||\cdot||$. La matrice identité de rang s est notée Id_s. On note $I_{*,j}$ le vecteur colonne dont tous les éléments sont nuls, sauf celui situé à la ligne i qui a une valeur de 1. $AI_{*,j}$ correspond au j-ième vecteur colonne de A. En notant $I_{i,*} = I'_{*,i}$, on obtient que $I_{*,i}A$ correspond à la i-ième ligne de A. Par conséquent, on a

$$I_{i,*}AI_{*,j} = a_{i,j}$$

6.4.2 Plongement dans un espace quadratique

On considère E un sous ensemble fini de points de $\mathcal{G}$ muni d'une fonction de dissimilarité w.

Définition 16 *Un plongement des données E muni de la dissimilarité w est un triplet (ψ, r, q) où q est une forme quadratique sur $\mathbb{R}^r$ et $\psi : E \to \mathbb{R}^r$ est une fonction telle que :*
- *l'espace engendré par l'ensemble des vecteurs $\psi(E) + z$ ($z \in \mathbb{R}^r$) est $\mathbb{R}^r$*
- *$w(x, y) = q(\psi(x) - \psi(y))$*

Le paramètre r est la dimension du plongement et le plongement est dit propre si la fonction ψ est injective, et régulière si q est non singulière.

Soit $E = \{x_0, x_1, \ldots, x_T\}$ un sous-ensemble fini de $T+1$ éléments de $\mathcal{G}$. Soit c un élément de E. On cherche à définir un plongement de E pour la dissimilarité w. Sans perdre en généralité, on considère $c = x_0$. Soit $\psi : E \to \mathbb{R}^T$ la fonction définie par $\psi(x_0) = 0$ et $\psi(x_k) = I_{*,k}$ pour tout $k = 1..T$. Soit $M(w, c) = (m_{i,j})_{1 \leq i,j \leq T}$ la matrice carrée de dimension $T \times T$ telle que pour tout $0 \leq i, j \leq T$,

$$m_{i,j} = \frac{w(c, x_i) + w(c, x_j) - w(x_i, x_j)}{2}$$

La matrice $M(w, c)$, notée M dans la suite, est symétrique et induit donc une forme bilinéaire symétrique $< X|Y >= X'MY$ sur $\mathbb{R}^T$. On peut remarquer que $m_{i,0} = m_{0,i} = 0$ pour tout $i = 0..T$, ce qui justifie d'ignorer l'index 0 et de considérer M comme une matrice de dimension $T \times T$. Soit q la forme quadratique associée à $< \cdot|\cdot >$. Par construction et polarisation, on a :

$$< I_{*,i}|I_{*,j} >= m_{i,j}$$
$$q(I_{*,i}) = w(c, x_i)$$
$$q(I_{*,i} - I_{*,j}) = q(I_{*,i}) + q(I_{*,j}) - 2 < I_{*,i}|I_{*,j} >$$

d'où $q(\psi(x_i) - \psi(x_j)) = w(x_i, x_j)$, ce qui montre qu'il est toujours possible de trouver un plongement. Dans la suite, nous nommerons cette solution le plongement canonique centré en c.

Conformément à la théorie des formes quadratiques réelles, on sait qu'il existe une base orthonormale $V = (V_1, \ldots, V_T)$ (par rapport à la structure euclidienne classique sur $\mathbb{R}^T$) de vecteurs (colonnes) propres comptant avec leur multiplicité correspondant aux valeurs propres λ_i de M. Cette base est orthogonale par rapport à $< \cdot|\cdot >$ mais $q(V_i) = \lambda_i$. Par convention, nous supposerons que les valeurs propres positives sont $\{\lambda_1, \ldots, \lambda_p\}$ et que les valeurs propres négatives sont $\{\lambda_{p+1}, \ldots, \lambda_{p+n}\}$. Les valeurs propres $\{\lambda_{p+n+1}, \ldots, \lambda_T\}$ sont nulles. Le couple (p, n) est appelé la signature de q (pour T fixé). On a

$$M = V \begin{pmatrix} \lambda_1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \lambda_i & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \lambda_T \end{pmatrix} V'$$

La signature (p, n), qui caractérise q sur $\mathbb{R}^T$ à un isomorphisme près, ne dépend pas du choix du point origine $c \in E$, ni du choix de l'ordre des points dans E. En effet, considérons une permutation μ des éléments telle que $x_i^* = x_{\mu(i)}$ et $< \cdot|\cdot >^*$ la forme bilinéaire associée à la matrice M^* obtenue avec les points x_i^*. On a

$$m_{i,j} =< I_{*,i}|I_{*,j} >^* =< I_{*,\mu(i)} - I_{*,\mu(0)}|I_{*,\mu(j)} - I_{*,\mu(0)} >^*$$

avec $I_{*,0}$ le vecteur nul. $< \cdot|\cdot >^*$ est donc la forme quadratique $< \cdot|\cdot >$ mais exprimée dans la base $\{I_{*,\mu(i)} - I_{*,\mu(0)}, 1 \leq i \leq T\}$.

Nous avons montré l'existence d'un plongement cependant, le plongement obtenu n'est d'aucune aide pour l'analyse de E et w car il ne possède aucune structure géométrique. Pour tirer un intérêt du plongement de l'ensemble E muni de la dissimilarité w, il nous parait donc nécessaire de trouver un plongement standard.

Définition 17 *Une forme quadratique q non singulière d'index de Sylvester (p, n) est définie (à une permutation près de coordonnées) pour tout $X = (x_1, \ldots, x_r)' \in \mathbb{R}^r$ par :*

$$q(X) = \sum_{i=1}^{p} x_i^2 - \sum_{j=1}^{n} x_{p+j}^2$$

Définition 18 *Un plongement standard de E muni de w de signature (p, n) est un plongement (ψ, r, q_n) tel que :*

- *r est l'entier positif le plus petit possible*
- *q_n est une forme quadratique standard non singulière d'index de Sylvester (p, n) donc $r = p + n$.*

La forme polaire associée à q_n est notée $< \cdot|\cdot >_n$ avec

$$< X|Y >_n = X'\Sigma(p, n)Y$$

et

$$\Sigma(p, n) = \begin{bmatrix} Id_p & 0 \\ 0 & -Id_n \end{bmatrix}$$

Pour la recherche d'un plongement standard, deux cas peuvent être distingués : le cas où $n = 0$ *i.e.* lorsque q_n est semi-définie positive et le cas où $n > 0$. Le premier cas correspond à un plongement standard dans un espace euclidien tandis que le deuxième cas correspond à un plongement standard dans un espace pseudo-euclidien.

6.4.3 Plongement dans un espace euclidien

Lorsque M est semi-définie positive, la matrice M peut se ré-écrire sous le forme

$$M = \mathcal{X}'\mathcal{X}$$

avec χ une matrice rectangulaire

$$\mathcal{X} = \begin{pmatrix} \sqrt{\lambda_1} & 0 & \ldots & \ldots & 0 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \sqrt{\lambda_i} & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \ldots & \ldots & 0 & \sqrt{\lambda_p} & 0 & \cdots & 0 \end{pmatrix} V'$$

Soit $\psi_c : E \to \mathbb{R}^p$ la fonction définie par

$$\psi_c(x_j) = \mathcal{X} I_{*,j} = \begin{bmatrix} \mathcal{X}_{1,j} \\ \vdots \\ \mathcal{X}_{p,j} \end{bmatrix}$$

et $\psi_c(c) = 0$. M est alors la matrice de Gram du vecteur ligne $\mathcal{X} = [\psi_c(x_1), \ldots, \psi_c(x_T)]$ *i.e.* $m_{i,j} = \psi_c(x_i)'\psi_c(x_j) = (\psi_c(x_i)|\psi_c(x_j))$. Par définition, $\mathcal{X}$ est de rang p. La fonction ψ_c plonge les données de E dans l'espace $\mathbb{R}^p$ de manière à ce que la propriété suivante soit vérifiée :

$$w(x_i, x_j) = q_0(\psi_c(x_i) - \psi_c(x_j)) = ||\psi_c(x_i) - \psi_c(x_j)||^2$$

Théorème 1 (ψ_c, p, q_0) *est un plongement standard.*

En suivant la méthodologie de l'analyse en composantes principales à noyaux (Schölkopf et al., 1999), la matrice duale $\mathcal{X}\mathcal{X}'$ introduit les axes principaux des vecteurs $\psi_c(E)$. Les axes obtenus à partir de la base orthonormale $[U_1, \ldots, U_p]$ (dans l'espace euclidien standard $\mathbb{R}^p$) sont donnés par

$$\begin{aligned} U_i &= \frac{1}{\sqrt{\lambda_i}} \mathcal{X} V_i \quad (i = 1, \ldots, p) \\ &= \frac{1}{\sqrt{\lambda_i}} \sum_{j=1}^{T} (V_i)_j \psi_c(x_j) \end{aligned} \tag{6.1}$$

avec $(V_i)_j$ le j-ième élément du vecteur colonne V_i. U_i est le vecteur propre normalisé de la matrice duale associée à la valeur propre λ_i. Par construction de $\mathcal{X}$, on a, pour tout $i = 1..p$,

$$\mathcal{X} V_i = \sqrt{\lambda_i} I_{*,i}$$

et

$$\mathcal{X}\mathcal{X}' = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$$

avec $\mathrm{diag}(\lambda_1, \ldots, \lambda_p)$ la matrice diagonale d'éléments diagonaux λ_1, ..., λ_p. Par conséquent, $[U_1, \ldots, U_p]$ est la base orthonormale standard de l'espace euclidien $\mathbb{R}^p$.

L'équation 6.1 permet d'introduire l'opérateur linéaire $\Gamma : \mathbb{R}^T \to \mathbb{R}^p$ défini par

$$\Gamma = \mathcal{X} \left[\sum_{i=1}^{p} \frac{1}{\sqrt{\lambda_i}} \Pi_i \right]$$

avec Π_i la projection orthogonale sur l'espace vectoriel engendrée par V_i *i.e.* $\Pi_i X = (X|V_i)V_i$. On a alors

$$\Gamma V_i = \begin{cases} U_i & \text{si } i = 1..p \\ 0 & \text{sinon} \end{cases}$$

La restriction de Γ à l'espace orthogonal $Ker(q)^\perp$ est une isométrie entre $Ker(q)^\perp$ et $\mathbb{R}^p$ tous les deux équipés de la structure euclidienne classique.

$\Gamma'\Gamma$ est le projecteur orthogonal sur l'espace vectoriel engendré par $[V_1, \ldots, V_p]$ *i.e.*

$$\Gamma'\Gamma = V \mathrm{diag}(signe(\lambda_1), \ldots, signe(\lambda_T)) V'$$

avec

$$signe(\lambda) = \begin{cases} 1 & \text{si } \lambda > 0 \\ 0 & \text{si } \lambda = 0 \\ -1 & \text{si } \lambda < 0 \end{cases}$$

La projection des vecteurs $\psi_c(x_k)$ sur l'axe U_i est donnée par

$$(\psi_c(x_k)|U_i) = \psi_c(x_k)'U_i$$

Le cas d'une forme quadratique de signature $(0, n)$ est similaire au cas euclidien, après avoir remplacé $w(\cdot,\cdot)$ par son opposé.

6.4.4 Plongement dans un espace pseudo-euclidien

Le troisième cas que nous considérons est le cas d'une forme quadratique de signature (p, n) avec $p > 0$ et $n > 0$.

On définit

$$D^+ = \begin{bmatrix} \sqrt{\lambda_1} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sqrt{\lambda_p} \end{bmatrix}$$

$$D^- = \begin{bmatrix} \sqrt{-\lambda_{p+1}} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \sqrt{-\lambda_{p+n}} \end{bmatrix}$$

Soit Δ^+ et Δ^- les matrices rectangulaires respectivement de dimensions $p \times T$ et $n \times T$ défini par

$$\Delta^+ = [D^+ \; 0_{[p\times(T-p)]}]$$

$$\Delta^- = [0_{[n\times p]} \; D^- \; 0_{[n\times(T-n-p)]}]$$

avec $0_{[a\times b]}$ la matrice nulle de dimension $a \times b$. Si $\Delta = \begin{bmatrix} \Delta^+ \\ \Delta^- \end{bmatrix}$ alors

$$\Delta'\Sigma(p,n)\Delta = \mathrm{diag}(\lambda_1, \dots, \lambda_T)$$

Soit $\mathcal{X} = \begin{bmatrix} \mathcal{X}^+ \\ \mathcal{X}^- \end{bmatrix} = \begin{bmatrix} \Delta^+V' \\ \Delta^-V' \end{bmatrix} = \Delta V'$. La matrice M vérifie $M = \mathcal{X}'\Sigma(p,n)\mathcal{X}$. Cette matrice peut être ré-écrite à l'aide de deux composantes : l'une « positive », l'autre « négative », telles que :

$$M = M^+ + M^-$$

$$M^+ = (\mathcal{X}^+)'\mathcal{X}^+$$

$$M^- = -(\mathcal{X}^-)'\mathcal{X}^-$$

On remarque que $M^+ - M^- = \mathcal{X}'\mathcal{X}$, $M^+M^- = M^-M^+ = 0$ et $\mathcal{X}'\mathcal{X}V_i = |\lambda_i|V_i$.

Les vecteurs propres $V_1, \dots, V_p$ sont des vecteurs propres de M^+ associés aux mêmes valeurs propres que M et les vecteurs propres $V_{p+1}, \dots, V_{p+n}$ sont des vecteurs propres de M^- associés aux valeurs $-\lambda_{p+1}, \dots, -\lambda_{p+n}$. Les autres vecteurs propres font partie du noyau de M^+ ou M^-. Les matrices M^+ et $-M^-$ sont donc symétriques semi définies positives.

De manière similaire au cas euclidien, on introduit la fonction $\psi_c : E \to \mathbb{R}^{p+n}$, telle que

$$\psi_c(x) = \begin{bmatrix} \psi_c(x)^+ \\ \psi_c(x)^- \end{bmatrix}$$

avec $\psi_c(x_i)^+ = \mathcal{X}^+I_{*,i}$ et $\psi_c(x_i)^- = \mathcal{X}^-I_{*,i}$ d'où $\psi_c(x_i) = \mathcal{X}I_{*,i}$ pour $i = 1..p+n$.

Nous munissons l'espace $\mathbb{R}^{p+n}$ de la structure pseudo-euclidienne donnée par la forme quadratique standard q_n de signature (p,n). La fonction ψ_c vérifie

$$< \psi_c(x_i)|\psi_c(x_j) >_n = I_{i,*}\mathcal{X}'\Sigma(p,n)\mathcal{X}I_{*,j}$$

d'où $w(x_i,x_j) = q_n(\psi_c(x_i) - \psi_c(x_j))$. De plus $\psi_c(E)$ engendre $\mathbb{R}^{p+n}$

Théorème 2 *$(\psi_c, p+n, q_n)$ est un plongement standard de E*

Comme précédemment, nous considérons la matrice duale $\mathcal{X}\mathcal{X}'$ comme un noyau dont les axes principaux sont fournis par

$$\begin{aligned} U_i &= \frac{1}{\sqrt{|\lambda_i|}}\mathcal{X}V_i,\ i = 1,\ldots,p+n \\ &= \frac{1}{\sqrt{|\lambda_i|}}\sum_{j=1}^{T}(V_i)_j\psi_c(x_j)\,. \end{aligned}$$

Par construction, on a $\mathcal{X}V_i = \sqrt{|\lambda_i|}I_{*,i}$ et $\mathcal{X}\mathcal{X}' = \mathrm{diag}(|\lambda_1|,\ldots,|\lambda_{p+n}|)$.
$U = \{U_1,\ldots,U_{p+n}\}$ est la base canonique de $\mathbb{R}^{p+n}$. En effet, on a

$$< U_i|U_j >= \begin{cases} signe(\lambda_i) & \text{si } i = j \\ 0 & \text{sinon} \end{cases}$$

L'opérateur linéaire $\Gamma : \mathbb{R}^T \to \mathbb{R}^{p+n}$ défini par

$$\Gamma = \mathcal{X}\left[\sum_{i=1}^{p+n}\frac{1}{\sqrt{|\lambda_i|}}\Pi_i\right]$$

vérifie

$$\Gamma V_i = \begin{cases} U_i & \text{si } i = 1..p+n \\ 0 & \text{sinon} \end{cases}$$

La restriction de Γ sur $Ker(q)^\perp$ est donc une isométrie de $Ker(q)^\perp$ sur $\mathbb{R}^{p+n}$. Les formules suivantes sont vérifiées :

$$\Gamma'\Sigma(p,n)\Gamma = V\mathrm{diag}(signe(\lambda_1),\ldots,signe(\lambda_T))V'$$

$$\Gamma\Gamma' = \mathrm{Id}_{p+n}$$

La décomposition de $\psi_c(x)$ dans la base U conduit à

$$\psi_c(x) = \sum_{i=1}^{p+n} signe(\lambda_i) < \psi_c(x)|U_i >_n U_i$$

avec

$$\begin{aligned} < \psi_c(x_k)|U_i >_n &= \psi_c(x_k)'\Sigma(p,n)U_i \\ &= I_{k,*}\mathcal{X}'\Sigma(p,n)\frac{1}{\sqrt{|\lambda_i|}}\mathcal{X}V_i \\ &= \frac{1}{\sqrt{|\lambda_i|}}I_{k,*}MV_i \\ &= \frac{1}{\sqrt{|\lambda_i|}}\sum_{j=1}^{T}(V_i)_j < \psi_c(x_k)|\psi_c(x_j) >_n \end{aligned}$$

L'ensemble des formules développées dans le cas pseudo-euclidien sont également valables dans le cas euclidien, *i.e.* quand $n = 0$, à la différence que l'espace « négatif » associé aux valeurs propres négatives est réduit au singleton $\{0\}$.

6.4.5 Centrage du nuage de points

La dernière étape nécessaire pour rendre utile ce plongement standard consiste à centrer le nuage de points de manière à exprimer le plongement de manière indépendante de l'origine c choisie. Pour cela, on considère E et un plongement standard (ψ_c, r, q) d'origine c. Soit $\mu = \frac{1}{T}\sum_{i=1}^{T}\psi_c(x_i)$ le centre du nuage formé par les vecteurs $\psi_c(x_i)$ pour $i = 1..T$. Si $\psi(x) = \psi_c(x) - \mu$ alors (ψ, r, q) est également un plongement standard de E. Supposons l'existence d'un objet abstrait ν tel que $\psi_c(\nu) = \mu$ *i.e.* $\psi(\nu) = 0$. On a :

$$w(x_i, x_j) = q_n(\psi_c(x_i) - \psi_c(x_j)) = q_n(\psi(x_i) - \mu - (\psi(x_j) - \mu)) = q_n(\psi(x_i) - \psi(x_j))$$

$$w(\nu, x_i) = q_n(\mu - \psi_c(x_i)) = q_n(\psi(x_i))$$

La matrice M associée à ψ est donnée par :

$$\begin{aligned} m_{i,j} &= < \psi(x_i)|\psi(x_j) > \\ &= < \psi_c(x_i) - \mu|\psi_c(x_j) - \mu > \\ &= \frac{\tilde{w}_i + \tilde{w}_j - w(x_i, x_j) - \tilde{w}}{2} \end{aligned}$$

avec $\tilde{w}_i = \frac{1}{T}\sum_{i=1}^{T} w(x_i, x_j)$ et $\tilde{w} = \frac{1}{T^2}\sum_{i=1}^{T}\sum_{j=1}^{T} w(x_i, x_j)$. On remarque que cette formule correspond aux relations classiques de Torgeson (Torgeson, 1965) appliquées dans le cas euclidien. On remarque également que la matrice M ne dépend que de la dissimilarité w, et qu'elle est indépendante de l'origine c choisie.

Le centrage du nuage de points peut avoir des effets inattendus lorsque q n'est pas définie positive *i.e.* $p \neq 0$ et $n \neq 0$. En effet, dans ce cas, le noyau $Ker(q)$ n'est pas réduit au singleton $\{0\}$. Prenons l'exemple suivant : $T = 2$, $r = 2$, $q(x) = x_1^2 - x_2^2$, $\sigma(q) = (1, 1)$ et $\psi_c(E) = \{(1\ 0)',\ (3\ 2)'\}$. Dans ce cas, $\mu = (2\ 1)'$ et $\psi(E) = \psi_c(E) - \mu$ est contenu dans la droite isotrope $\mathbb{R}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$. La matrice M associée est nulle et $(\psi, 2, q)$ n'est pas un plongement standard.

6.4.6 Unicité du plongement

Pour montrer l'unicité du plongement à un isomorphisme près, on considère deux plongements standards (ψ, r, q_n) et $(\psi^*, r^*, q^*_{n^*})$ de E respectivement dans $\mathbb{R}^r$ et $\mathbb{R}^{r^*}$. Soit $\sigma(q_n) = (p, n)$ et $\sigma(q^*_*) = (p^*, n^*)$. Puisque $w(x_i, x_j) = q_n(\psi(x_i) - \psi(x_j)) = q_{n^*}(\psi^*(x_i) - \psi^*(x_j))$, la translation de $\psi(E)$ n'a aucune influence sur la forme du nuage, donc nous pouvons supposer sans perte de généralité que les deux plongements standards (ψ, r, q_n) et $(\psi^*, r^*, q^*_{n^*})$ sont centrés. Puisqu'il s'agit de plongements standards, r et r^* sont minimaux d'où $r = r^*$. Soit $K = \{k_1, \ldots, k_r\} \subset E$ et $X_i = \psi(x_{k_i})$ tel que le vecteur ligne $X = [X_1, \ldots, X_r]$ forme une base de $\mathbb{R}^r$. Soit $X^* = [X_1^*, \ldots, X_r^*]$ le vecteur ligne tel que $X_i^* = \psi^*(x_{k_i})$.

Par définition, la matrice de Gram $G_n(X) = X'\Sigma(p, n)X$ associée à X conformément à q_n est égale à la matrice de Gram $G^*_{n^*}(X^*) = X^{*\prime}\Sigma(p^*, n^*)X^*$ associée à X^* conformément à q^*_n. Par conséquent, X^* est également une base de $\mathbb{R}^r$ et il existe une matrice de dimension $r \times r$ telle que $AX = X^*$. Par construction, pour tout $(X, Y) \in (\mathbb{R}^r)^2$, on a $< AX|AY >_{n^*} = < X|Y >_n$ donc A est un isomorphisme entre $(\mathbb{R}^r, q_n)$ et $(\mathbb{R}^r, q^*_{n^*})$. Cela implique que $p = p^*$ et $n = n^*$. Puisque les vecteurs $\psi(x_i)$ et $\psi^*(x_i)$ sont déterminés par les coordonnées $< \psi(x_i)|X_j >_n$ et $< \psi^*(x_i)|X_j^* >_{n^*}$ sur les bases X et X^*, on a $\psi^* = A \circ \psi$. L'ensemble de ces conclusions nous permettent d'établir le théorème suivant.

Théorème 3 *Soit (ψ, r, q_n) et $(\psi^*, r^*, q^*_{n^*})$ deux plongements standards de E muni de la dissimilarité w. Soit G et G^* les matrices de Gram obtenues à partir de $\psi(E)$ et $\psi^*(E)$ dans les espaces quadratiques $(\mathbb{R}^r, q_n)$ et $(\mathbb{R}^{r^*}, q^*_{n^*})$. Dans ces conditions, on a :*

- *$r = r^* = rang(G) = rang(G^*)$*
- *q_n et $q^*_{n^*}$ sont identiques et de signature égale à celle de G et G^**
- *il existe deux translations τ_1 et τ_2 sur $\mathbb{R}^r$ et un automorphisme A de l'espace quadratique $(\mathbb{R}^r, q)$ tel que $\psi^* = (\tau_1 \circ A \circ \tau_1) \circ \psi$*

En considérant toujours les notations précédentes, la matrice M se décompose sous la forme $M = \mathcal{X}'\Sigma(p,n)\mathcal{X}$. Soit $\mathcal{X}^* = [\psi^*(x_1), \dots \psi^*(x_T)]$ la matrice associée à ψ^*. Par construction, on a $\mathcal{X}^{*\prime}\Sigma(p,n)\mathcal{X}^* = M$. En appliquant le théorème précédent, on sait qu'il existe une matrice carrée de dimension $r \times r$ telle que $A'\Sigma(p,n)A = \Sigma(p,n)$ d'où $\mathcal{X}^* = A\mathcal{X}$. On définit $\mathcal{X}^{*+} = A\mathcal{X}^+$ et $\mathcal{X}^{*-} = A\mathcal{X}^-$. On a $M^* = M^{*+} + M^{*-}$ avec $M^{*+} = \mathcal{X}^{*+\prime}\Sigma(p,n)\mathcal{X}^{*+} = M^+$ et $M^{*-} = \mathcal{X}^{*-\prime}\Sigma(p,n)\mathcal{X}^{*-} = M^-$ donc la décomposition de M ne change pas avec le plongement.

Les axes du nuage associés à la matrice duale $\mathcal{X}^*\mathcal{X}^{*\prime} = A\mathcal{X}\mathcal{X}A'$ sont donnés par $U^*_i = AU_i = AI_{*,i}$. La base $U^* = AU$ est orthonormale par rapport au pseudo produit scalaire $< \cdot|\cdot >_n$ *i.e.*

$$< U^*_i | U^*_j >_n = \begin{cases} signe(\lambda_i) & \text{si } i = j \\ 0 & \text{sinon} \end{cases}$$

D'où

$$\mathcal{X}^*\mathcal{X}^{*\prime} = A\text{diag}(|\lambda_1|, \dots, |\lambda_{p+n}|)A'$$

L'opérateur Γ^* correspondant à Γ vérifie $\Gamma^* = A\Gamma$. La formule

$$\Gamma^{*\prime}\Sigma(p,n)\Gamma^* = V\text{diag}(signe(\lambda_1), \dots, signe(\lambda_T))V'$$

est toujours valide mais $\Gamma^*\Gamma^{*\prime} = AA'$

L'ensemble de ces développements nous ont permis de montrer qu'il était toujours possible de plonger un ensemble de données muni d'une dissimilarité dans un espace pseudo-euclidien de dimension minimale. La construction est obtenue à partir d'une adaptation de la technique de l'analyse en composantes principales à noyau (*Kernel PCA*) dans laquelle le noyau n'est pas forcément défini positif. Nous avons nommé cette décompostion l'Analyse en Composantes Principales à Noyau Indéfini (ACPNI) (*Indefinite Kernel PCA* ou IKPCA en langue anglaise). La normalisation induite par l'opérateur Γ permet de réaliser une analyse du nuage de façon similaire à celle que l'on peut réaliser avec une analyse en composantes principales à noyaux dans un espace euclidien.

La méthode que nous proposons peut être re-cadrée naturellement dans la structure en deux phases introduite à la section 6.2. En effet, le plongement des données dans un espace pseudo-euclidien correspond à la phase de représentation tandis que la recherche d'un plongement standard centré et le calcul des axes principaux correspond à la phase de projection.

6.5 La matrice de *scatterplots* pseudo-euclidienne

Finalement, dans la deuxième partie de cette section, nous présenterons les modifications nécessaires pour adapter la technique de la matrice de *scatterplots* à la visualisation du nuage (Aupetit et al., 2005c) (Aupetit et al., 2005b).

6.5.1 Caractéristiques des espaces pseudo-euclidiens

L'utilisation de l'ACPNI pour la visualisation du nuage nécessite, dans un premier temps, de mieux connaître les espaces pseudo-euclidiens.

Le cas le plus simple correspond à un espace pseudo-euclidien $(\mathcal{E}, q)$ standard de dimension 1. Que la signature de q soit $(1,0)$ ou $(0,1)$, le cône isotrope $C(q)$ est réduit au singleton $\{0\}$. Par conséquent, les représentants $\psi(x)$ et $\psi(y)$ de deux objets x et y similaires ont des coordonnées proches.

Supposons maintenant que $(\mathcal{E}, q)$ est un espace pseudo-euclidien standard de dimension 2. La signature de q peut être $(2,0)$, $(1,1)$ ou $(0,2)$.

Dans le premier cas $\sigma(q) = (2,0)$, $(\mathcal{E}, q)$ est un espace euclidien, dans lequel deux objets x et y sont similaires si leurs images $\psi(x)$ et $\psi(y)$ ont des coordonnées proches et réciproquement. De plus, quel que soit $x \in \mathcal{E}$, on a $q(c) \geq 0$. Les isovaleurs $q(x) = c$ pour $c \geq 0$ (correspondant à l'ensemble des points ayant une dissimilarité constante c par rapport au centre de l'espace) sont des cercles. De plus, le cône isotrope $C(q)$ est réduit au singleton $\{0\}$. La figure 6.2 permet de visualiser ces isovaleurs et le cône isotrope.

Dans le deuxième cas $\sigma(q) = (0,2)$, on remarque que $(\mathcal{E}, -q)$ est du même type que le premier cas, donc deux objets x et y sont similaires si leurs images $\psi(x)$ et $\psi(y)$ ont des coordonnées proches et réciproquement. La notion de similarité de x et y est, dans ce cas, évaluée à partir de $-w(x,y)$ au lieu de $w(x,y)$. De plus, quel que soit $x \in \mathcal{E}$, $q(x) \leq 0$, les isovaleurs $q(x) = c$ pour $c \leq 0$ (correspondant à l'ensemble des points ayant une dissimilarité constante c par rapport au centre de l'espace) sont des cercles. De plus, le cône isotrope $C(q)$ est réduit au singleton $\{0\}$. La figure 6.3 permet de visualiser ces isovaleurs et le cône isotrope.

Le dernier cas $\sigma(q) = (1,1)$ correspond au plan hyperbolique. Dans ce plan, le cône isotrope $C(q)$ est l'union des lignes $\binom{1}{1}\mathbb{R}$ et $\binom{1}{-1}\mathbb{R}$. De plus, il existe $x \in \mathcal{E}$ et $y \in \mathcal{E}$ tels que $q(x) > 0$ et $q(y) < 0$. Ce plan peut également être étudié avec une structure euclidienne. On remarque alors que, pout tout $\epsilon > 0$, il existe $(x,y) \in \mathcal{E}^2$ tels que $||x-y|| \leq \epsilon$ et $q(x-y) \geq 1$. De plus, il existe $(x,y) \in \mathcal{E}^2$ tels que $||x-y|| > 0$ et $q(x-y) = 0$. Ces propriétés sont perturbantes mais ce sont ces dernières qui font qu'un plongement standard existe toujours. Les isovaleurs correspondant à $q(x) = c$ et le cône isotrope sont fournis par la figure 6.4.

Les espaces euclidiens de dimension supérieure à 2 restent difficiles à étudier mais tout espace $\mathcal{E}$, qu'il soit euclidien ou pseudo-euclidien, peut être décomposé sous la forme d'une somme directe d'espaces de dimension inférieure à 2 :

$$\mathcal{E} = \mathcal{P}_1 \oplus \cdots \oplus \mathcal{P}_k$$

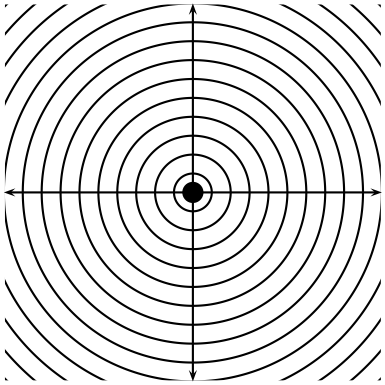
ou

$$\mathcal{E} = \mathcal{P}_1 \oplus \cdots \oplus \mathcal{P}_k \oplus \mathcal{E}_*$$

avec $\mathcal{P}_i$ un plan pseudo-euclidien de dimension 2 et $\mathcal{E}_*$ un espace euclidien ou anti-euclidien de signature $(1,0)$ ou $(0,1)$.

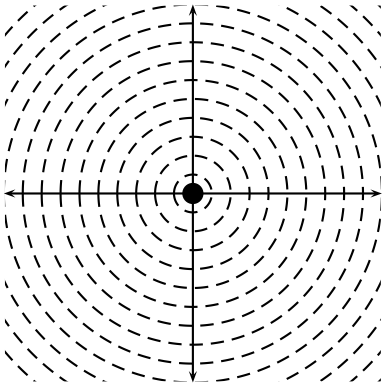
6.5.2 Adaptation de la matrice de *scatterplots* pour la visualisation dans un espace pseudo-euclidien

La technique de la matrice de *scatterplots* permet la visualisation du nuage $\psi(E)$ sous la forme de graphe réalisant des projections du nuage $\psi(E)$ selon deux de ses coordonnées. La matrice de *scatterplots* est composée de l'ensemble des graphes possibles (Fayyad et al., 2001) (Wong and Bergeron, 1997). L'un des principaux avantages de cette technique est que chaque variable (*i.e.* coordonnée) joue le même rôle dans la visualisation (Wijk and van Liere, 1993). Les matrices de *scatterplots* sont utilisées principalement pour répondre à trois objectifs (Fayyad et al., 2001) : détecter des corrélations entre variables, détecter des groupes et détecter des points aberrants.



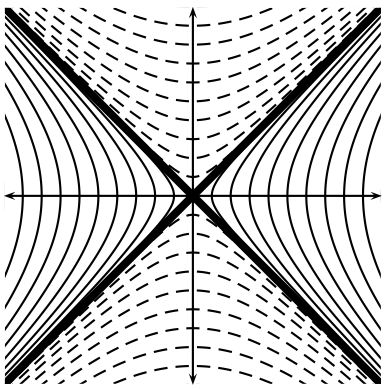
Les isovaleurs correspondant à des valeurs c positives sont en trait plein, celles correspondant à des valeurs c négatives sont en trait pointillé et le cône isotrope est dessiné en trait épais.

FIG. 6.2 – Isovaleurs d'un espace pseudo-euclidien de dimension 2, lorsque $\sigma(q) = (2, 0)$.



Les isovaleurs correspondant à des valeurs c positives sont en trait plein, celles correspondant à des valeurs c négatives sont en trait pointillé et le cône isotrope est dessiné en trait épais.

FIG. 6.3 – Isovaleurs d'un espace pseudo-euclidien de dimension 2, lorsque $\sigma(q) = (0, 2)$.



Les isovaleurs correspondant à des valeurs c positives sont en trait plein, celles correspondant à des valeurs c négatives sont en trait pointillé et le cône isotrope est dessiné en trait épais.

FIG. 6.4 – Isovaleurs d'un espace pseudo-euclidien de dimension 2, lorsque $\sigma(q) = (0, 2)$.

Du fait de la construction des coordonnées du nuage $\psi(E)$, la détection de corrélation entre les dimensions est peu intéressante. Cependant, la détection et/ou la confirmation de groupes et la détection de points aberrants restent intéressantes pour l'étude du nuage. Ces actions auraient pu être effectuées par des techniques moins complexes mais, dans le cas présent, l'information d'échelle dans les groupes et le positionnement relatif des groupes, les uns par rapport aux autres sont directement accessibles.

Si on utilise directement les coordonnées $\psi(E)$ dans une matrice de *scatterplots*, l'interprétation risque d'être erronée, car la matrice de *scatterplots* permet de visualiser naturellement un nuage dans un espace euclidien, et non pas dans un espace pseudo-euclidien. Ainsi, en fonction du type de sous-espaces de dimension 2, deux points visuellement proches peuvent correspondre à des objets très dissemblables ou non et inversement deux points visuellement éloignés peuvent correspondre à des objets semblables ou non.

6.5.2.1 Isovaleurs

Afin de permettre l'interprétation de ces sous-espaces de dimension 2, nous proposons d'utiliser les isovaleurs des figures 6.2, 6.3 et 6.4. Ainsi, lorsque l'utilisateur clique avec la souris à un endroit p du plan, on dessine les isolignes centrées sur p en fonction de la signature du sous-espace de dimension 2. Le clic sur l'image du point x_i dans ce plan permet de déterminer visuellement la dissimilarité entre x_i et les autres points de E dans le plan considéré. On pourra remarquer qu'il ne s'agit pas de la dissimilarité réelle w entre deux objets x et y, mais uniquement de la contribution à la dissimilarité sur ce plan. Ainsi, si $1, \ldots, k$ représentent les dimensions de l'espace $\mathcal{E}$, alors la dissimilarité apparente sur le plan correspondant aux dimensions i et j pour les objets x et y est

$$signe(\lambda_i)(\psi(x)_i - \psi(y)_i)^2 + signe(\lambda_j)(\psi(x)_j - \psi(y)_j)^2$$

et non pas $q(\psi(x) - \psi(y))$. Pour bien analyser l'espace, il est donc nécessaire de considérer les plans associés aux graphes de manière à ce qu'une dimension ne soit pas utilisée deux fois. Il serait donc possible de réduire le nombre de graphes de la matrice de *scatterplots* mais en les laissant tous, l'utilisateur reste libre du choix des plans qu'il considère.

6.5.2.2 Taille des graphes

Dans une matrice de *scatterplots* classique, les différents graphes ont la même taille, donc l'étendue des valeurs prises par les points dans une dimension est normalisée dans $[0; 1]$. Cette normalisation a pour but de ne privilégier aucune dimension par rapport aux autres. Si l'on applique ce procédé à la visualisation d'un plongement dans un espace pseudo-euclidien, on introduit une distorsion des formes quadratiques de chacun des plans. Cette distorsion a pour effet de gommer les différences entre les dimensions et surtout de modifier l'angle (visuel) formé par les axes du cône isotrope dans le cas $\sigma(1, 1)$, et de transformer les isovaleurs en ellipse, dans le cas $\sigma(q) = (2, 0)$ ou $\sigma(q) = (0, 2)$. Ainsi, pour chaque graphe, la structure du plan change. Afin d'éviter ce problème et de faciliter l'interprétation des données, nous avons choisi de dimensionner les graphes de manière à ce que la forme quadratique $q_{i,j}$ associée aux dimensions i et j soit de la forme

$$q_{i,j}(x) = \pm x_i^2 \pm x_j^2$$

Si l'on note β_i l'amplitude des valeurs prises par les poins $\psi(E)$ pour la dimension i, alors le plan associé aux dimensions i et j est mis à l'échelle

$$\left(\frac{\beta_i}{\max_k \beta_k}, \frac{\beta_j}{\max_k \beta_k}\right)$$

dans la construction de la matrice. Sous ces conditions, les isolignes associées à des espaces de même signature ont la même forme. Afin de faciliter l'interprétation, les isovaleurs sont colorées dans le modèle couleur RVB par $(\max(0, 1-c), 0, 0)$, où c est la valeur des isovaleurs.

6.5.2.3 Remplissage

Les données sont représentées dans les graphes par des carrés pouvant être remplis de différentes façons :

- par une image associée à la donnée
- par une couleur calculée à partir d'une valeur numérique, ordinale, symbolique ou calendaire[6] associée à la donnée. L'utilisateur peut alors définir un ensemble de couples (valeur,couleur) utilisés pour le remplissage. Entre deux plages de valeurs définies par l'utilisateur, la couleur est obtenue par interpolation linéaire dans l'espace de couleur Rouge Vert Bleu (RVB). Soit $(val_i, coul_i)$ pour $i = 1..K$ les couples tels que $val_i \leq val_{i+1}$. La couleur obtenue pour une valeur val est

$$coul(val) = \begin{cases} coul_1 & \text{si } val \leq val_1 \\ coul_i + \frac{val - val_i}{val_{i+1} - val_i}(coul_{i+1} - coul_i) & \text{si } val_i \leq val \leq val_{i+1} \\ coul_K & \text{si } val \geq val_K \end{cases}$$

- par une couleur calculée à partir de trois valeurs associées à la donnée. Le principe de calcul est similaire au précédent, mais il s'effectue sur chacune des composantes RVB.

Le remplissage par couleurs permet de comprendre l'influence de certaines caractéristiques des objets visualisés sur la dissimilarité. Cependant, les données n'ont pas forcément de caractéristiques utiles.

6.5.2.4 Marquage

Pour faciliter la détection des groupes et de points aberrants, nous avons ajouté la possibilité de marquer les points par un cadre de couleur. Le marquage d'un point dans un plan permet de marquer de manière simultanée l'image de ce point dans tous les plans de la matrice de *scatterplots*. Le marquage des points est réalisé selon un principe similaire à celui décrit dans (Becker and Cleveland, 1987). La sélection des points candidats au marquage s'effectue à la souris en dessinant un rectangle sur un graphe. Cependant, contrairement au procédé décrit dans (Becker and Cleveland, 1987), le marquage des points peut ne pas correspondre à une zone rectangulaire. En effet, en raison de la structure de l'espace, des objets similaires peuvent ne pas être contenus dans une sélection rectangulaire. Les points marqués sont donc ceux contenus dans l'union de zones rectangulaires. Un double clic de souris sur un objet permet de marquer un seul objet afin d'afficher dans une table les attributs correspondant à l'objet. Ce marquage spécifique permet de faire le lien entre les groupes, la dissimilarité et les attributs des objets.

6.5.2.5 Sélection des axes principaux

L'ACPNI de T objets de E crée un plongement (ψ, r, q) tel que $r \leq T$. La dimension r de l'espace dans lequel s'effectue ce plongement et donc le nombre d'axes principaux associés à une valeur propre non nulle peut être très grand. Or la taille d'un écran est limitée et, même en utilisant l'interaction pour voir une surface d'écran plus grande, la matrice de *scatterplots* perd de son intérêt et de son utilité au fur et à mesure que r augmente. Un moyen de réduire

[6] Les valeurs ordinales et symboliques sont converties en nombres entiers et les valeurs calendaires sont converties en nombre de jours du calendrier Julien/Grégorien (nombre de jours depuis le 1er janvier de l'an -4712, 12h depuis le méridien de Greenwich).

le nombre d'axes principaux utilisés pour la visualisation consiste à ne sélectionner que les axes les plus significatifs pour l'analyse. Dans une analyse en composantes principales (Bertier and Bouroche, 1975) (Escofier and Pagès, 1998), les axes les plus significatifs correspondent aux valeurs propres les plus grandes (Harris, 1985). Dans le cas de l'ACPNI, nous proposons d'utiliser la magnitude des valeurs propres comme critère de sélection. Soient $\lambda_1^+, \ldots, \lambda_p^+$ les p valeurs propres positives associées à q en ordre décroissant *i.e.* $\lambda_1^+ \geq \ldots \geq \lambda_p^+$. Soient λ_1^-, $\ldots$, λ_n^- les n valeurs propres négatives associées à q en ordre décroissant de magnitude *i.e.* $|\lambda_1^-| \geq \ldots \geq |\lambda_n^-|$. La sélection des k^+ et k^- axes principaux associés aux valeurs propres positives et négatives de plus grande amplitude permet de définir quatre indicateurs pour évaluer cette sélection :

$$\eta^+ = \frac{\sum_{i=1}^{k^+} \lambda_i^+}{\sum_{i=1}^{p} \lambda_i^+}$$

$$\eta^- = \frac{\sum_{i=1}^{k^-} |\lambda_i^-|}{\sum_{i=1}^{n} |\lambda_i^-|}$$

$$\eta^* = \frac{\sum_{i=1}^{k^+} \lambda_i^+ + \sum_{i=1}^{k^-} |\lambda_i^-|}{\sum_{i=1}^{p} \lambda_i^+ + \sum_{i=1}^{n} |\lambda_i^-|}$$

$$\eta = \frac{\sum_{i=1}^{n} |\lambda_i^-|}{\sum_{i=1}^{p} \lambda_i^+}$$

η^+ et η^- mesurent respectivement la quantité d'information effectivement sélectionnée dans les sous espaces « positifs » et « négatifs ». η^* mesure la quantité d'information effectivement sélectionnée. Finalement, η mesure la quantité d'information initialement portée par l'espace « négatif » par rapport à l'espace « positif ». Lorsque η a une valeur importante (ex : 0.3), cela signifie que l'espace « négatif » n'est pas négligeable par rapport à l'espace « positif » dans la contribution à la dissimilarité. Lorsque η est proche de 0, cela signifie que seuls les axes principaux associés à des valeurs propres positives sont utiles. Dans le cas inverse, la visualisation doit prendre en compte des axes principaux « positifs » et « négatifs ». La sélection des axes principaux peut donc s'effectuer en prenant ceux associés aux valeurs propres (positives et négatives) de magnitude les plus grandes, tandis que la validation de cette sélection peut s'effectuer à l'aide de ces quatre indicateurs.

La technique de visualisation que nous venons de proposer porte le nom de matrice de *scatterplots* pseudo-euclidienne (MSPE).

6.6 Applications

Afin de montrer l'intérêt de la technique de la MSPE, nous l'avons appliquée à l'étude de plusieurs dissimilarités sur deux types de données : la base Iris (Fisher, 1936) et la base ORL (Samaria and Harter, 1994).

6.6.1 Analyse de dissimilarités sur la base Iris

La base Iris (Blake and Merz, 1998) (Fisher, 1936) est une base de données classiquement utilisées pour comparer et mettre en évidences les caractéristiques des techniques de visualisation. Comme son nom l'indique, cette base de données décrit des mesures effectués sur des Iris (la plante). Elle est composée de 150 points de $\mathbb{R}^4$ répartis dans trois classes de même taille. La première classe est linéairement séparable des deux autres tandis que ces deux dernières ne le sont pas l'une de l'autre (cf. figure 6.6). Pour appliquer le MSPE, nous avons choisi de considérer cinq dissimilarités, afin de voir leur impact sur ces données. Les dissimilarités sont :

$$
\begin{aligned}
w_{\text{abs}}(x,y) &= \sum_{i=1}^{4} |x_i - y_i| \\
w_2(x,y) &= \sqrt{\sum_{i=1}^{4} (x_i - y_i)^2} \\
w_{\max}(x,y) &= \max_{i=1}^{4} |x_i - y_i| \\
w_{\text{threshold}}(x,y) &= \begin{cases} 0 & \text{if } w_2(x,y) < 1 \\ w_2(x,y) - 1 & \text{if } w_2(x,y) \geq 1 \end{cases} \\
w_{\text{exp}}(x,y) &= 1 - exp(-\frac{w_2(x,y)^2}{2})
\end{aligned}
$$

Les figures 6.5, 6.6, 6.7, 6.8 et 6.9 présentent les matrices de *scatterplots* pseudo-euclidiennes après coloration des points en fonction de leurs classes. Les axes principaux choisis, ainsi que les valeurs des quatre indicateurs, sont donnés par le tableau 6.1.

Dissimilarité	η	η^+	η^-	η^*	Valeur propre
w_{abs}	10%	82%			1743
		89%			160
			25%		−54
		92%		85%	49
w_2	0%	92%			630
			97%	97%	36
$w_{\max}$	6%	90%			455
		92%			13
			26%		−8
		93%		90%	7
$w_{\text{threshold}}$	34%	96%			17327
			81%		−5032
		99%			520
			89%	96%	−477
w_{exp}	9%	50%			26
		73%			12
		82%			5
		87%		87%	2

Les valeurs du tableau sont arrondies à l'entier le plus proche.

TAB. 6.1 – Axes principaux et indicateurs correspondant aux figures 6.5, 6.6, 6.7, 6.8 et 6.9.

Comme nous pouvons le voir, quelle que soit la dissimilarité que nous considérons, avec quatre axes principaux au maximum, on a $\eta \geq 80\%$. Par conséquent, il est raisonnable de considérer que les figures 6.5, 6.6, 6.7, 6.8 et 6.9 sont représentatives de l'organisation induite par les dissimilarités. Pour chacune de ces matrices, on remarque que le premier axe correspond à la séparation de la classe la plus isolée (en vert) des deux autres, donc chacune de ces dissimilarités respecte la tendance générale dans les données. On remarque également que les données faisant partie de la même classe sont localisées dans des zones bien définies de l'espace. Par conséquent, ces dissimilarités préservent les groupes et pourraient donc être utilisées pour effectuer une détection de groupe ou de la classification. Cependant, la forme des groupes et la séparation entre les groupes varient en fonction de la dissimilarité. La possibilité de choisir une dissimilarité, plutôt qu'une autre, permet donc, en fonction des besoins, d'utiliser les propriétés de conservation de l'organisation globale des données, tout en ayant la liberté de choix sur la forme des classes.

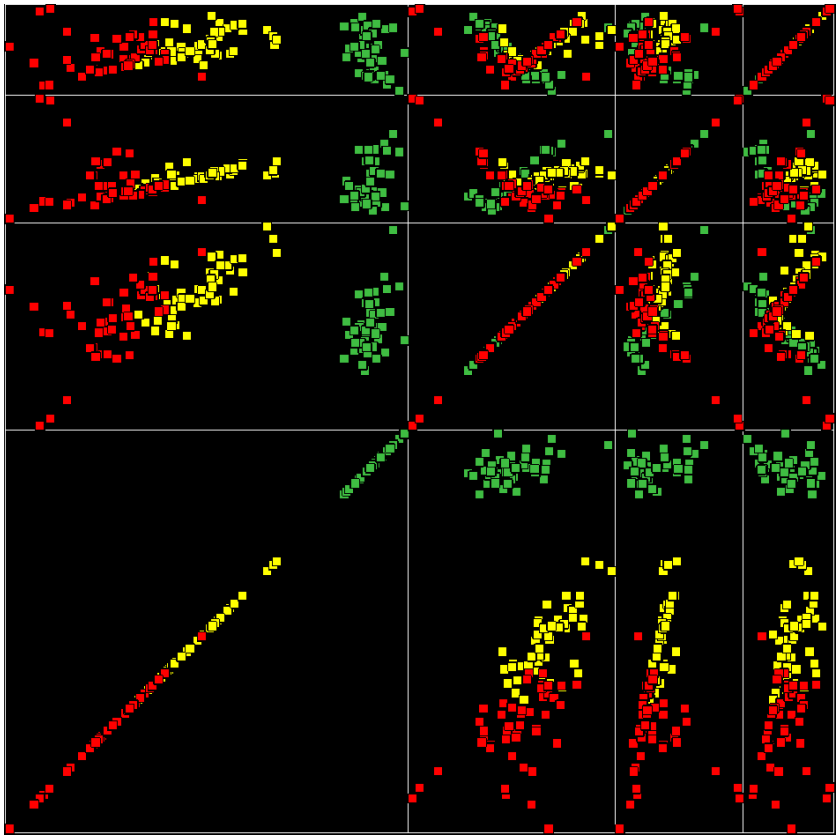


FIG. 6.5 – MSPE des 4 premiers axes principaux pour w_{abs}.

6.6.2 Analyse de dissimilarités sur des modèles de Markov cachés

Dans de nombreuses applications des modèles de Markov cachés, il serait utile d'avoir à disposition une dissimilarité, afin de pouvoir comparer les modèles entre eux. Il existe de nombreuses dissimilarités pour des MMC (cf. 4.5.1.5 p. 100), telles que la distance de Kullback-Leibler ou tout simplement la distance euclidienne entre les matrices des modèles. Supposons que l'on cherche à comparer deux MMC ayant le même nombre d'états et le même nombre de symboles. Certaines de ces dissimilarités ont pour inconvénient de comparer l'état s_i d'un MMC avec l'état s_i de l'autre MMC. Or les MMC sont connus à la renumérotation près des états, donc la comparaison effectuée ne reflète pas les capacités des MMC. Dans (Rabiner, 1989), l'auteur évoque une dissimilarité ne posant pas ce problème, mais elle nécessite

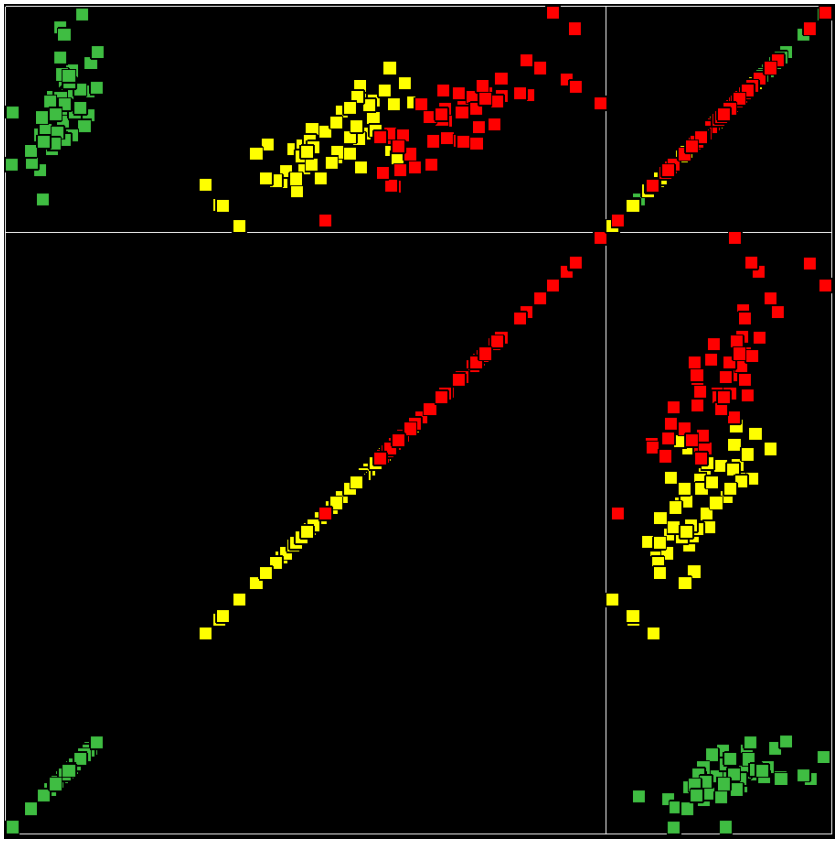


FIG. 6.6 – MSPE des 2 premiers axes principaux pour w_2.

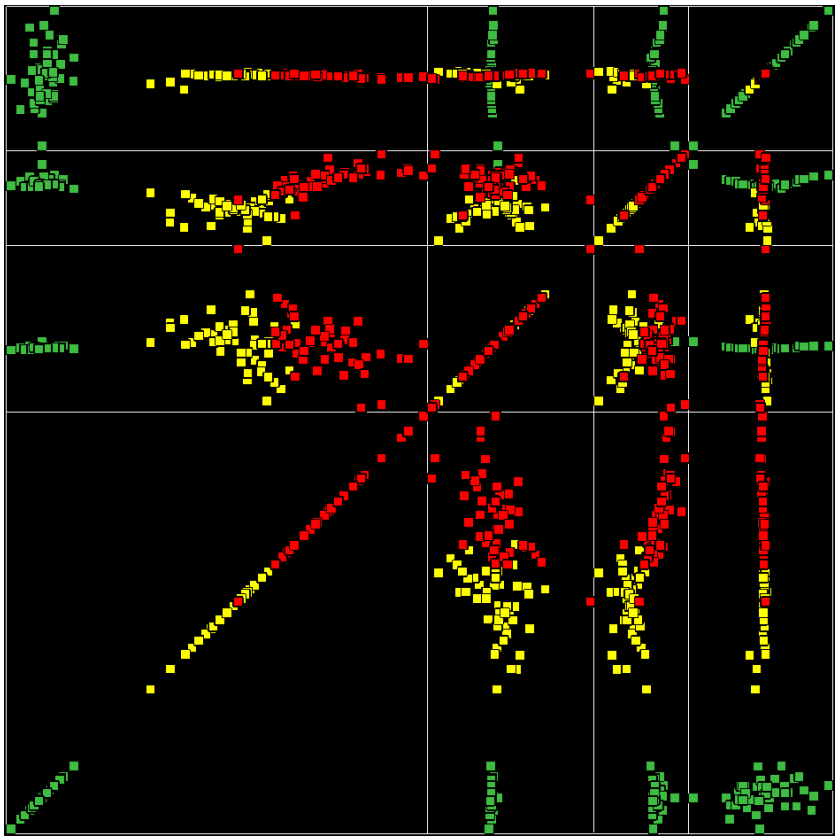


FIG. 6.7 – MSPE des 4 premiers axes principaux pour $w_{\max}$.

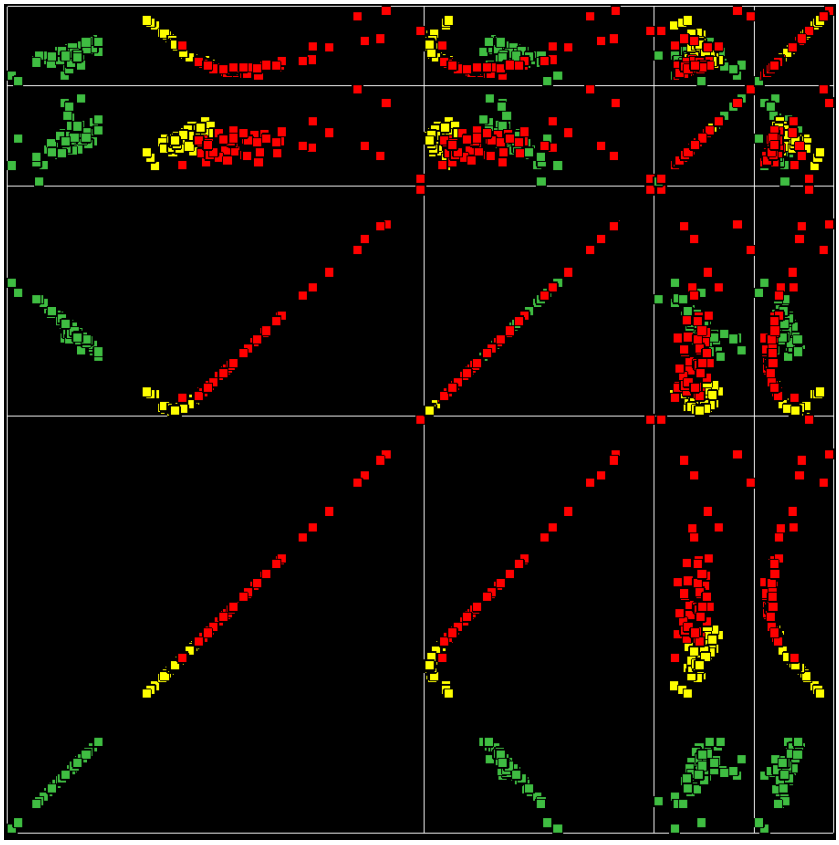


FIG. 6.8 – MSPE des 4 premiers axes principaux pour $w_{\text{threshold}}$.

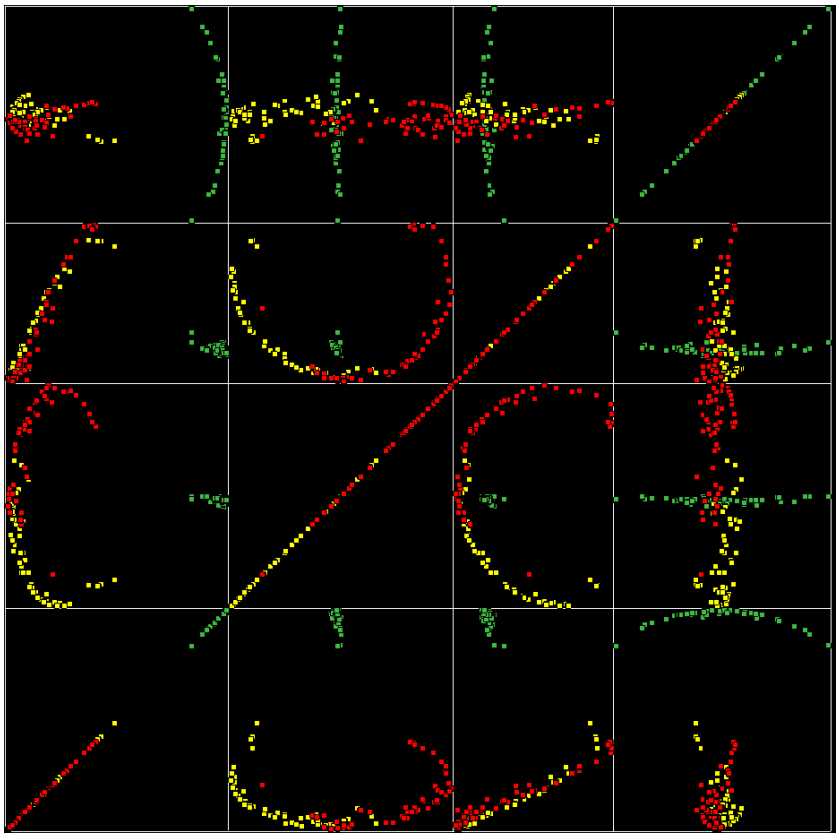


FIG. 6.9 – MSPE des 4 premiers axes principaux pour w_{exp}.

d'engendrer en probabilité des séquences d'observations. Par conséquent, en fonction des séquences engendrées, la valeur de la dissimilarité peut varier. Pour réduire cette variation, il est nécessaire d'effectuer de nombreux tirages aléatoires, donc ce calcul devient très rapidement coûteux. La distance de Kullback-Leibler (Falkhausen et al., 1995) possède les mêmes inconvénients : pour une estimation robuste, de nombreux tirages aléatoires sont nécessaires. De plus, les approximations rapides existantes, comme celles données dans (Do, 2003) et (Vihola et al., 2002), mesurent les états les uns par rapport aux autres. Comme nous pouvons le voir, ces dissimilarités sont trop coûteuses à calculer ou inadaptées[7] à la comparaison de MMC.

Face à ce constat, nous avons cherché à définir une dissimilarité capable de comparer des MMC, sans faire jouer un rôle particulier aux états correspondants dans les deux modèles (Aupetit et al., 2005c). Dans cette optique, nous souhaitons comparer les distributions de probabilités des symboles engendrées par les modèles. Pour les définir, nous considérons une distribution particulière des probabilités des états cachés à un instant donné. Nous nottons $\gamma = (\gamma_1, \ldots, \gamma_N)'$ cette distribution de probabilités des états cachés. Nous avons considéré deux distributions particulières des états cachés :

- La distribution d'équilibre ou d'incertitude : $\gamma_e = (1/N, \ldots, 1/N)'$. Le MMC peut être dans n'importe quel état avec la même probabilité.
- La distribution limite à l'infinie γ_∞, définie comme étant la limite de la suite $A^t\Pi$. Cette limite converge si tous les coefficients de A sont non nuls. Dans le cas contraire, cette suite peut converger vers différentes distributions limites. Dans ce cas, on en choisit une arbitrairement.

Pour construire les dissimilarités, nous pouvons considérer deux types de distributions paramétrées par λ et γ :

- les distributions d'ordre 1 des symboles :

$$\mu_{\lambda,\gamma}(k) = \sum_{i=1}^{N} \gamma_i b_i(k)$$

- les distributions d'ordre 2 des symboles (apparition de deux symboles à la suite)

$$\rho_{\lambda,\gamma}(k,l) = \sum_{i=1}^{N}\sum_{j=1}^{N} \gamma_i b_i(k) a_{i,j} b_j(l)$$

Ces distributions permettent de construire les dissimilarités suivantes :

$$w^{(1)}_{\gamma_e}(x,y) = MRE(\mu_{x,\gamma_e(x)}, \mu_{y,\gamma_e(y)})$$

$$w^{(2)}_{\gamma_e}(x,y) = MRE(\rho_{x,\gamma_e(x)}, \rho_{y,\gamma_e(y)})$$

$$w^{(1)}_{\gamma_\infty}(x,y) = MRE(\mu_{x,\gamma_\infty(x)}, \mu_{y,\gamma_\infty(y)})$$

$$w^{(2)}_{\gamma_\infty}(x,y) = MRE(\rho_{x,\gamma_\infty(x)}, \rho y, \gamma_\infty(y))$$

en désignant par

$$MRE(u,v) = \frac{1}{2}\left(\sum_i u_i \ln\frac{u_i}{v_i} + \sum_i v_i \ln\frac{v_i}{u_i}\right)$$

la moyenne de l'entropie relative des distributions u et v.

[7] Nous entendons « inadaptées » dans le sens où la comparaison des MMC se fait état par état.

Pour visualiser l'effet de ces dissimilarités sur des MMC, on considère les 10 premiers visages des 5 premières personnes de la base ORL (Samaria and Harter, 1994) (cf. section 4.5.1.1 p. 88). Les images sont apprises par l'algorithme API avec la configuration de paramètres API-Hete2 (cf. section 4) après avoir été recodée en 64 niveaux de gris. Les MSPE obtenues sont données par les figures 6.10, 6.11, 6.12 et 6.13. La description des axes de l'analyse est donnée par la table 6.2. On remarque que dans les quatre cas, quatre axes sont suffisants pour avoir $\eta^* \geq 70\%$, donc les MSPE sont relativement assez fidèles au nuage d'origine.

On remarque que les valeurs propres des dissimilarités du premier ordre sont de même rapport par rapport à la somme des modules des valeurs propres. En effet, pour chacun des axes correspondant, l'explication η^+ ou η^- est quasiment identique. De plus, on remarque que les projections du nuage sur les plans sont quasiment identiques aux symétries près[8]. Grâce à ces propriétés, vérifiées aussi bien dans le cas de γ_e que de γ_∞ (sur les deux premiers axes), nous pouvons conclure que le coût calculatoire induit par les distributions d'ordre 2 n'apporte globalement rien de plus à la structure du nuage.

Lorsque les points sont remplis par les visages associés aux MMC, on s'aperçoit que le premier axe effectue la séparation des visages en fonction de la luminosité globale des images. De plus, on remarque la formation d'un coude dans le *scatterplot* correspondant aux deux premières valeurs propres. Or ce coude correspond au cône isotropique de l'espace (de signature $(1,1)$). Par conséquent, les deux premiers axes permettent de séparer les visages en deux groupes : ceux globalement clairs et ceux globalement sombres. Dans chacun de ces groupes, la contribution à la dissimilarité selon ces deux axes est nulle ou quasi-nulle. Par conséquent, l'effet de la dissimilarité peut être étudiée uniquement sur les deux derniers axes en considérant les groupes séparément.

On remarque alors que les visages sont bien regroupés pour chacune des personnes, donc ces dissimilarités réalisent une bonne séparation des personnes. Cependant, on peut remarquer qu'un MMC (en vert) pour la MSPE $w_{\gamma_e}^{(1)}$ se détache du bon comportement obtenu pour les autres MMC alors que pour $w_{\gamma_\infty}^{(1)}$ cette différence n'apparaît pas. Deux hypothèses peuvent alors être formulées : $w_{\gamma_\infty}^{(1)}$ est plus robuste que $w_{\gamma_e}^{(1)}$, car il n'accentue pas le bruit ou le MMC hors norme et le visage associé est un mauvais modèle du visage. En effectuant un nouvel apprentissage des MMC, on s'aperçoit qu'un point hors norme apparaît ou non quel que soit $w_{\gamma_e}^{(1)}$ ou $w_{\gamma_\infty}^{(1)}$. Par conséquent, il paraît raisonnable de considérer que le point hors norme est dû à l'apprentissage et au fait que le modèle obtenu n'est pas idéal.

Si on devait effectuer un choix parmi ces deux dissimilarités, il serait judicieux de considérer $w_{\gamma_e}^{(1)}$ plutôt que $w_{\gamma_\infty}^{(1)}$, car le coefficient η associé est plus petit, donc la dissimilarité est plus sujette à bien se comporter avec les algorithmes qui l'utiliseront.

On peut finalement remarquer que, comme précédemment (cf. section 6.6.1) les MMC associés à une personne sont géographiquement regroupés dans les *scatterplots*.

6.7 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle méthode de visualisation de dissimilarité utilisant la notion d'espace pseudo-euclidien, afin de plonger le nuage dans un espace vectoriel. Cette méthode appelée Matrice de *Scatterplots* Pseudo-Euclidienne (MSPE) est obtenue à partir de la technique classique de la matrice de *scatterplots* et d'une analyse en composantes principales utilisant des noyaux indéfinis (ACPNI). Nous avons montré que cette ACPNI correspondait à une analyse en composantes principales (ACP) ordinaire lorsque le noyau est défini positif. Cette méthode possède pour principal avantage d'être générique et

[8] Considérer la symétrie verticale et/ou horizontale est équivalent à considérer le vecteur propre $-u$ au lieu de u pour le support d'un axe. Par conséquent, ces symétries ne changent pas l'analyse du nuage.

Dissimilarité	η	η^+	η^-	η^*	Valeur propre
$w_{\gamma_e}^{(1)}$	50%	57%			8
			55%		-4
		73%			2
		80%		71%	1
$w_{\gamma_e}^{(2)}$	36%	59%			20
			55%		-7
		73%			5
		81%		74%	3
$w_{\gamma_\infty}^{(1)}$	62%	63%			90
			62%		-55
		78%			21
			76%	78%	-13
$w_{\gamma_\infty}^{(2)}$	55%	67%			170
			61%		-84
		78%			25
		85%		76%	18

Les valeurs du tableau sont arrondies à l'entier le plus proche.

TAB. 6.2 – Axes principaux et indicateurs correspondant aux figures 6.10, 6.11, 6.12 et 6.13.

de permettre la visualisation de n'importe quelle dissimilarité. Finalement, nous avons montré sur deux exemples que le choix d'une dissimilarité pouvait avoir de grandes conséquences sur la forme des classes d'objets. Nous avons également montré comment cette visualisation pouvait être utilisée pour comparer plusieurs dissimilarités.

Cette visualisation trouve donc sa place parmi les outils de l'analyse de données. Cependant, cette technique possède, comme toutes les autres, plusieurs limitations. La première limitation est due à l'ACPNI. En effet, comme pour une ACP, les axes principaux sont très influencés par le bruit dans les données, donc l'analyse peut être faussée par la présence de bruit dans les dissimilarités. La dimension de l'affichage implique que le nombre d'axes utilisables dans la MSPE ne doit pas être trop important. Pour que la représentation soit fiable, il est donc nécessaire que η^* soit suffisamment grand. Or, il est possible de trouver des dissimilarités ne permettant pas d'obtenir le compromis : η^* grand pour un nombre réduit d'axes. Un autre inconvénient dû à la méthode est que le nombre d'objets qui peut être considéré ne doit pas être trop grand pour ne pas surcharger la visualisation, et donc la rendre inutilisable. Finalement, le dernier inconvénient et peut être le plus important, est que cette méthode reste assez difficile à utiliser, car les notions utilisées (espaces pseudo-euclidiens) nécessitent une certaine habitude de la méthode.

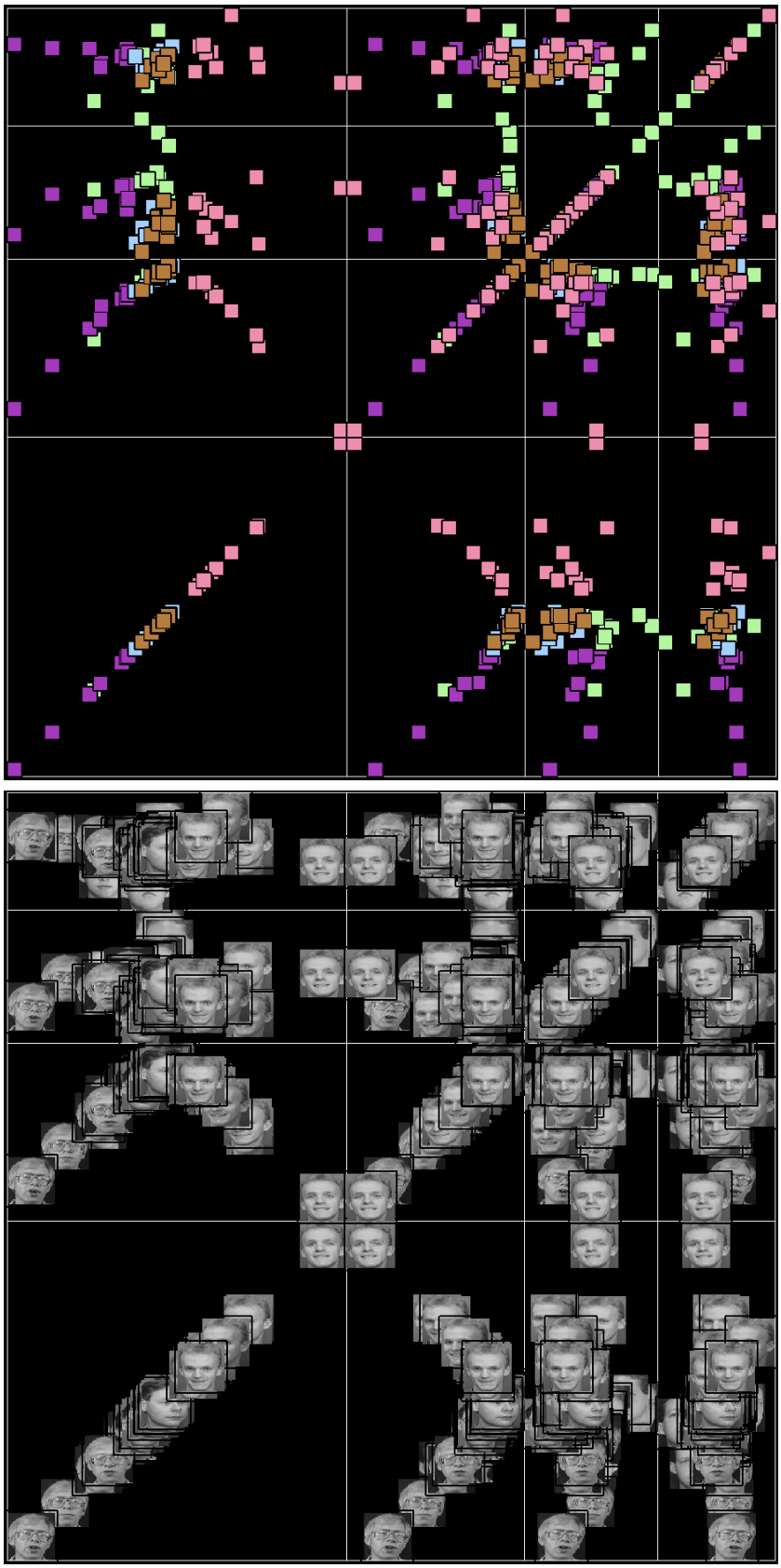


FIG. 6.10 – MSPE de $w_{\gamma_e}^{(1)}$ avec remplissage en couleur en fonction de la personne et avec remplissage par les visages associés.

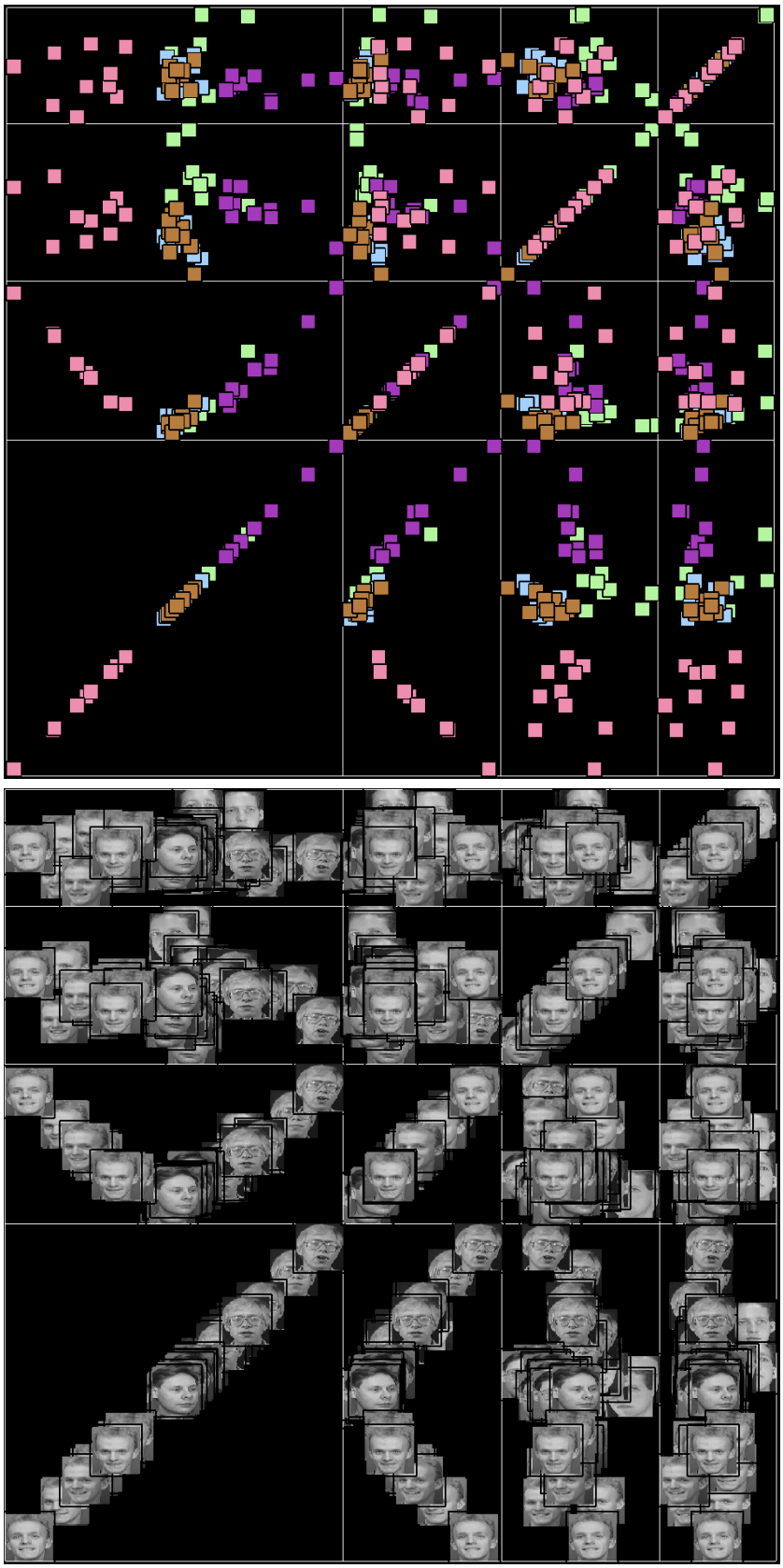


FIG. 6.11 – MSPE de $w_{\gamma_e}^{(2)}$ avec remplissage en couleur en fonction de la personne et avec remplissage par les visages associés.

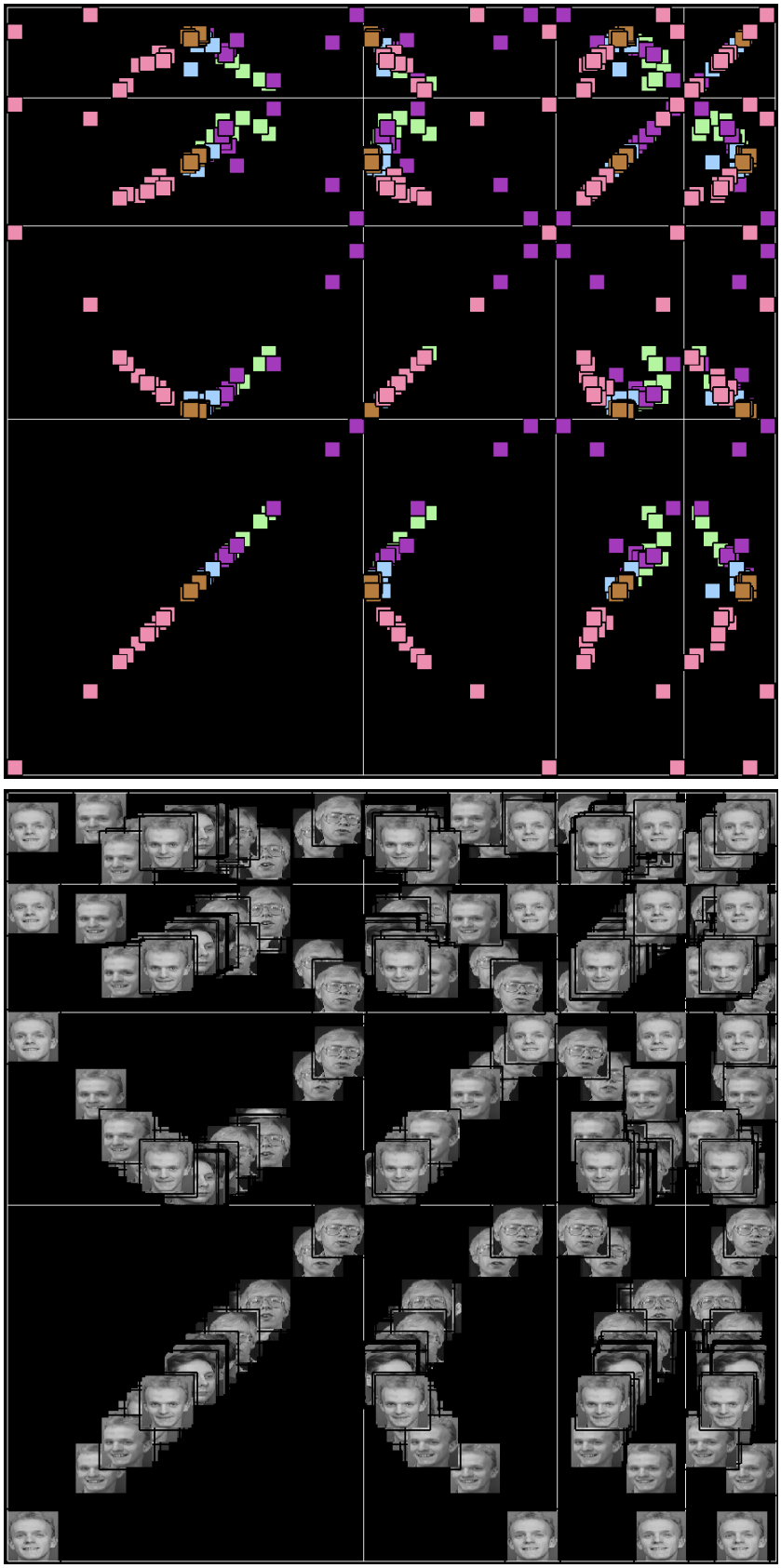


FIG. 6.12 – MSPE de $w_{\gamma_\infty}^{(1)}$ avec remplissage en couleur en fonction de la personne et avec remplissage par les visages associés.

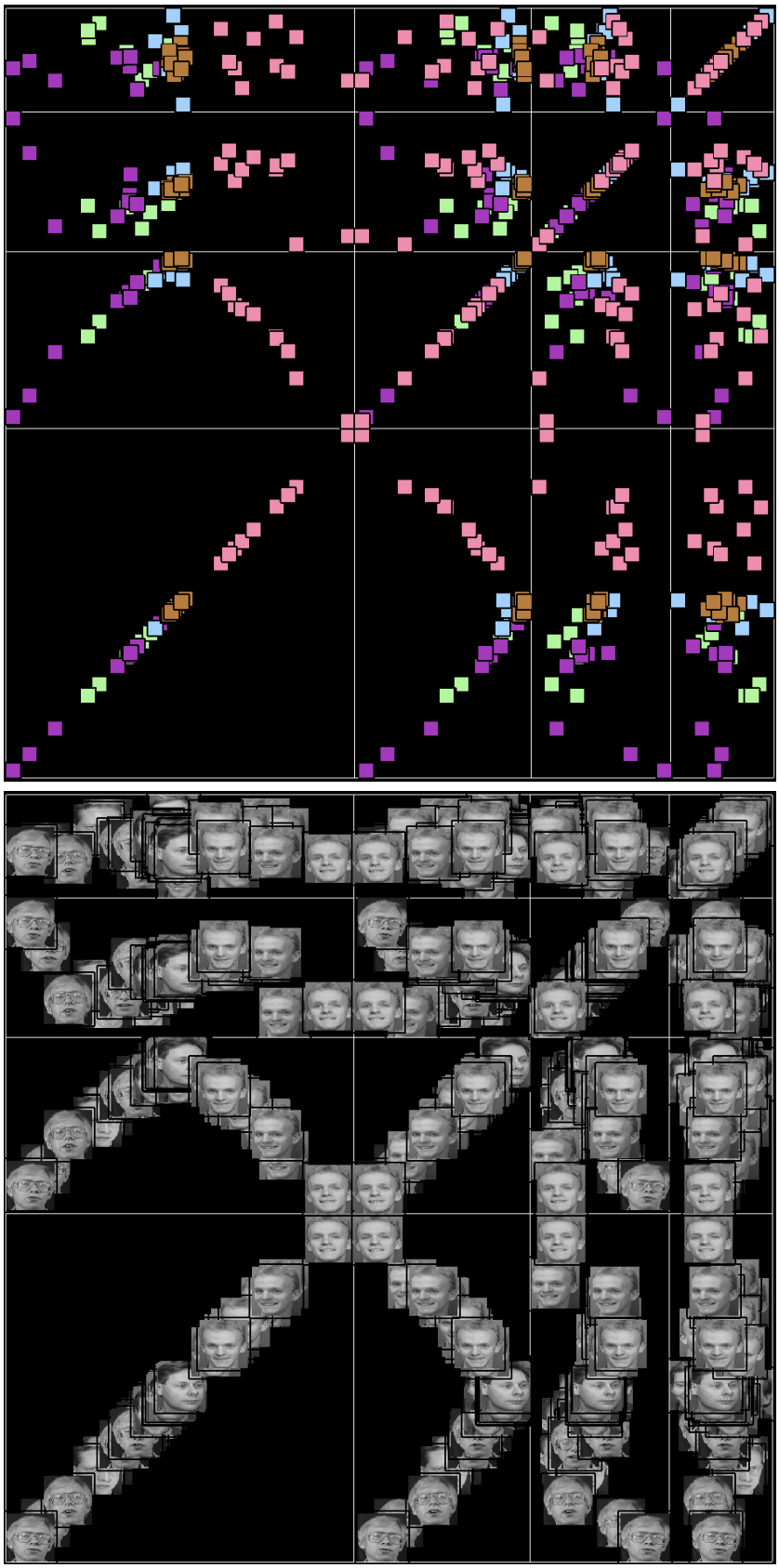


FIG. 6.13 – MSPE de $w_{\gamma_\infty}^{(2)}$ avec remplissage en couleur en fonction de la personne et avec remplissage par les visages associés.

Chapitre 7

HMMTK : *Hidden Markov Model Tool Kit*

7.1 Pourquoi une nouvelle bibliothèque ?

Au cours de cette thèse, il nous a été nécessaire d'utiliser les différents algorithmes liés aux modèles de Markov cachés et d'en implémenter d'autres. Le langage de programmation que nous avons décidé d'utiliser est le C++.

Au Laboratoire d'Informatique, dans une première tentative, Thierry Brouard (Brouard, 1999) a développé une bibliothèque de MMC. Nous avons donc décidé d'ajouter les travaux présentés dans cette thèse à cette bibliothèque. Cependant, la bibliothèque de T. Brouard pose un certain nombre de problèmes :

- l'ajout de fonctionnalités et d'algorithmes à la bibliothèque est relativement complexe et peu fiable ;
- la documentation de la bibliothèque est inexistante ;
- bien qu'une approche objet ait été employée pour son élaboration, elle a été poussée à l'extrême ce qui rend la bibliothèque difficile d'emploi ;
- aucun mécanisme ne permet de répartir sur plusieurs machines les nombreux calculs parfois nécessaires lorsqu'on effectue des expérimentations ;
- aucun mécanisme de vérification (mode débuggage) n'est prévu.

Suite à ces constatations, nous nous sommes orientés vers les autres bibliothèques de modèles de Markov cachés disponibles sur internet. Nous avons premièrement constaté que leur nombre et leurs fonctionnalités sont relativement réduits. Parmi les plus connues, on trouve la bibliothèque HTK (HTK, 2005) et la bibliothèque ghmm (GHMM, 2005). Ces deux bibliothèques, comme la plupart des autres, posent plusieurs problèmes :

- certaines s'adressent à des domaines très spécifiques, tels que la reconnaissance vocale (ex : HTK (HTK, 2005)) ;
- certaines n'en sont encore qu'au stade expérimental (ex : ghmm (GHMM, 2005)) ;
- certaines ne sont destinées qu'à des langages autres que le C++, tel que MATHLAB ;
- certaines possédent des licences pouvant poser des problèmes à terme (ex : HTK (HTK, 2005)).

Suite à ces investigations, nous avons décidé qu'il serait plus efficace de ré-implémenter une bibliothèque de modèles de Markov cachés qui corresponde à nos besoins.

Pour concevoir cette bibliothèque, nous nous sommes fixé plusieurs objectifs :

- la bibliothèque doit être généraliste, *i.e.* indépendante du domaine d'application ;
- la bibliothèque doit être modulaire, de manière à permettre de réaliser facilement des extensions et de gérer de nouveaux types de modèles ou des domaines d'applications particuliers, sans remettre en cause son architecture ;

- la bibliothèque doit intégrer des mécanismes de « débuggage » qui ne nuisent pas à la performance d'exécution ;
- la bibliothèque doit être sous licence libre (CeCILL v2) ;
- la bibliothèque doit permettre à moindre coût de paralléliser facilement les calculs sur plusieurs machines de manière quasi transparente à l'utilisateur ;
- la bibliothèque doit être d'utilisation simple.

La bibliothèque que nous avons développée a subi de nombreux changements jusqu'à sa stabilisation. La version actuelle est la version 5.0.0. Cette version satisfait les différents objectifs que nous nous sommes fixés et est maintenant suffisamment stable pour pouvoir être rendue publique. Les travaux présentés dans ce chapitre ont donné lieu à l'encadrement (Monvoisin and Viaux, 2005) (Navillat and Marty, 2005) (Bavoua and Boujnah, 2005) (Desmarest and Badille, 2005) (Chesnet and Herault, 2005) ou au co-encadrement (Lesné and Mahnich, 2003) (Martin and Pérotin, 2003) (Lesné and Perotin, 2004b) de plusieurs mini-projets visant à mettre en place l'environnement logiciel nécessaire à la parallélisation des calculs. Dans les sections suivantes, nous allons présenter les différents choix techniques et méthodologiques retenus lors de sa conception.

7.2 Méthodologie de conception

Afin de faciliter l'utilisation de la bibliothèque et permettre l'implémentation aisée de la parallélisation, nous avons adopté un paradigme de conception objet particulier. Dans ce paradigme, les données et les fonctions parallélisables sont des objets. Considérons un exemple simple. Soit f une fonction parallélisable de paramètres d'entrée E et de paramètres de sortie S. Pour transférer l'exécution de cette fonction de la machine X à la machine Y, il est nécessaire de suivre la procédure suivante (cf figure 7.1) :

- transférer E de la machine X vers la machine Y ;
- sur la machine Y, exécuter F avec les données E ;
- transférer le résultat S obtenu sur Y vers la machine X.

FIG. 7.1 – Principe du transfert de l'exécution d'une fonction $S = f(E)$.

Pour généraliser le processus de transfert de l'exécution, il est donc nécessaire de savoir transférer E et S d'une machine à une autre et de savoir exécuter F. Un moyen d'obtenir cette abstraction quels que soient E, S et F consiste donc à considérer l'ensemble comme un tout, c'est-à-dire comme une implémentation particulière d'une interface générique. Tout naturellement (mais pas de façon optimale), le transfert de données peut s'effectuer à travers de flux connectant une machine à une autre. Cette interface générique expose cinq méthodes principales, qu'il est nécessaire d'implémenter pour définir une fonction parallélisable :

- deux méthodes permettant de lire et écrire les paramètres S sur un flux ;
- deux méthodes permettant de lire et écrire les paramètres E sur un flux ;
- une méthode sans argument réalisant les calculs : les paramètres E et S sont alors des attributs membres de l'objet.

La figure 7.2 représente le modèle objet associé à la fonction parallélisable « algorithme *Forward* ». Pour cette fonction, les paramètres d'entrée sont *Observation* et *HMM*, tandis que les paramètres de sortie sont *LogProba*, *AlphaTilde* et *RescalingCoefs*. Les méthodes *ReadInput*,

WriteInput, *ReadOutput* et *WriteOutput* sont re-définies de manière à gérer automatiquement le transfert des paramètres d'entrée et de sortie sur un flux : écriture à un bout du flux et lecture à l'autre bout. Finalement la méthode *Run* est définie afin d'effectuer les calculs.

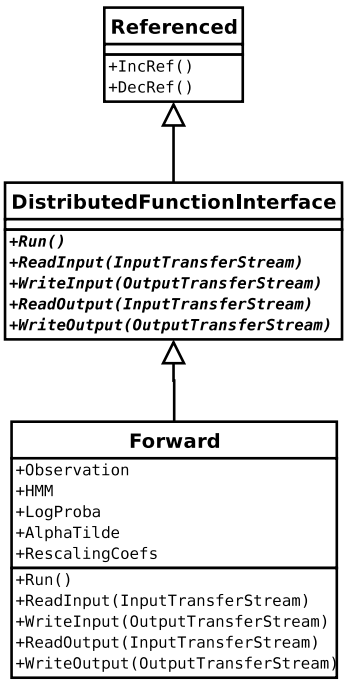


FIG. 7.2 – La fonction parallélisable « algorithme Forward ».

Les fonctions parallélisables et les flux sont les seuls mécanismes de parallélisation implémentés pour le moment dans la bibliothèque, pour la simple raison qu'ils sont suffisants dans les cas qui nous concernent. En effet, on remarque qu'en grande partie, les algorithmes (notamment les métaheuristiques) effectuent de nombreux calculs similaires et indépendants séquentiellement. Une méthode de parallélisation triviale consiste alors à effectuer ces calculs similaires mais indépendants en parallèle sur plusieurs machines. Nous considérons donc une parallélisation en « arbre ». Cette approche par décomposition permet d'économiser de nombreux transferts réseaux (synchronisations interprocessus, ...). De manière à effectuer ce type de parallélisation, nous avons défini un objet *Parallelizer* destiné à la parallélisation de tâches indépendantes. L'approche objet des fonctions parallélisables décrite plus haut se justifie encore car cela permet de manipuler des instances de paramètres d'entrée et de sortie des fonctions. Ces instances sont alors ajoutées dans une liste propre au *Parallelizer*. Par l'appel de la méthode *Run* du *Parallelizer*, ce dernier s'occupe, suivant le mode de parallélisation choisi, de paralléliser les calculs sur les différentes machines. Plusieurs modes de parallélisation seront décrits dans la section suivante.

Afin d'améliorer les performances de la bibliothèque, deux stratégies sont utilisées :

- afin d'éviter de nombreuses recopies de données en mémoire et par la même occasion pour réduire les temps d'exécution et la consommation mémoire, les structures de données et les fonctions parallélisables sont gérées via des pointeurs et des compteurs de référence. Du point de vue de l'utilisateur, il est important d'être attentif lors de l'utilisation de ces données « partagées », mais cela simplifie également la programmation et améliore la robustesse car les libération de mémoires sont gérées de manière automatique et transparente.
- l'utilisateur de la bibliothèque a la possibilité de limiter la profondeur de parallélisation, afin de ne pas surcharger les machines avec un nombre trop important de processus. Cette possibilité est très intéressante. Prenons un exemple, supposons que l'exécution

d'une tâche se décompose en trois tâches A, B et C. Chacune de ces tâches se décompose en trois sous-tâches : A1, A2, A3, B1, B2, B3, C1, C2 et C3. Si la décomposition continue, on peut se trouver face à la parallélisation de très nombreuses tâches, ce qui conduirait à une surcharge des machines et donc à une réduction de l'efficacité de la parallélisation. Un moyen simple que nous avons envisagé pour contrôler la charge en nombre de tâches est de limiter la profondeur de décomposition : une profondeur de 0 empêche la décomposition, une profondeur de 1 ne produit que les tâches A, B et C, ...

7.3 Méthode de parallélisation

Comme il a été évoqué précédemment, plusieurs modes de parallélisation ont été implémentés au fil des versions de la bibliothèque. Le mode de parallélisation est défini à la compilation de la bibliothèque, de manière à éviter les incohérences, mais également de manière à optimiser les performances.

7.3.1 Mode non parallélisé

Lorsque nous avons conçu la bibliothèque HMMTK, nous avons décidé que la parallélisation des algorithmes était nécessaire mais, pour ne pas la limiter, nous avons également décidé qu'elle devait pouvoir fonctionner de manière identique (du point de vue de l'utilisateur de la bibliothèque) lorsque la parallélisation n'était pas requise. A cet effet, deux options sont possibles :

- régler la profondeur de décomposition à 0 : mais cela nécessite de modifier une variable dans le programme, donc il ne s'agit pas totalement d'une désactivation transparente de la parallélisation
- implémenter l'objet *Parallelizer* de manière à ce qu'il effectue les calculs qui lui sont confiés séquentiellement. Cette stratégie ne nécessite aucune modification des programmes de calcul. L'unique modification se situe au moment de la configuration de la bibliothèque HMMTK. Depuis la version 5.0.0, la bibliothèque définit une constante supplémentaire du préprocesseur, qui, si elle est activée (selon les choix de l'utilisateur), permet d'optimiser encore davantage certaines portions de codes[1] au sein de HMMTK, en évitant la création de plusieurs fonctions parallélisables en mémoire et l'utilisation des *Parallelizer* au sein des métaheuristiques.

7.3.2 Mode parallélisé

En mode parallélisé, nous avons fait le choix de paralléliser des processus. La parallélisation aurait également pu être faite à l'aide de *threads* mais plusieurs raisons nous ont poussés vers la solution processus :

- la programmation par *threads* est beaucoup plus complexe que la programmation par processus ;
- les bibliothèques de parallélisation, telles que MPI (MPI, 2005) ou PVM (PVM, 2005) sont orientées processus ;
- si l'on souhaite effectuer un équilibrage de la charge des machines par un système tel que *OpenMosix* (OpenMosix, 2005), il est nécessaire d'avoir des processus *mono-threadés* pour qu'il soit pris en compte dans l'équilibrage (migration de processus d'une machine à une autre).

[1] Actuellement, seul le module DiscretHMM exploite cette fonctionnalité.

Lorsque la bibliothèque est configurée en mode parallélisé, plusieurs types de communications et de créations de processus sont utilisables : MPI, *shell* ou client-serveur. Pour chacun de ces modes de communication, aucun moyen n'est disponible pour gérer les machines disponibles. Nous avons donc constitué un *daemon* gérant les ressources.

7.3.2.1 *Daemon* de parallélisation

Le *daemon* de parallélisation possède un fonctionnement simple et possède très peu de fonctionnalités. Pour communiquer avec le *daemon*, les processus de calcul et le processus de l'utilisateur se connectent via un *socket* bidirectionnel au *daemon*. Les trois actions principales qui peuvent être demandées au *daemon* sont :

- L'allocation et la libération d'un emplacement sur une machine pour un processus. Afin de répartir au mieux les processus sur les machines, nous avons choisi d'affecter un processus à la machine la moins utilisée dans le sens du nombre actuel de processus divisé par le nombre maximal de processus autorisés sur cette machine.
- L'arrêt de tous les processus de calcul et du *daemon*.
- L'enregistrement et l'interrogation des informations nécessaires à la création des processus sur chacune des machines (ex : le chemin vers l'exécutable, ...).

7.3.2.2 Mode MPI2

Le premier mode de communication et de création des processus est le mode MPI2. Dans ce mode, les communications entre les processus s'effectuent grâce aux fonctions de communication de MPI (MPI, 2005). Pour permettre la création à la demande de processus sur les hôtes de la machine virtuelle MPI, la *commande MPI_ Comm_ spawn* est utilisée. Cette commande fait partie de la deuxième version du standard MPI dite MPI2. Par conséquent, pour utiliser ce mode de parallélisation, il est nécessaire de disposer d'une implémentation MPI2 de la norme. Actuellement, peu d'implémentations fiables de MPI2 sont disponibles gratuitement. A ce jour, la bibliothèque HMMTK a été testée uniquement avec l'implémentation LAM/MPI (LAM/MPI, 2005). Afin de paralléliser au mieux l'ensemble des tâches $\mathcal{T} = \{t_1, \ldots, t_T\}$ par MPI tout en prenant en compte les contraintes liées à MPI, la fonction Run de l'objet *Parallelizer* est défini par l'algorithme 7.1.

Dans cet algorithme, le lecteur remarquera que l'attente de la fin d'une tâche est bloquant et non interruptible, ce qui peut poser des problèmes de temps processeur inutilisé si les tâches ont des temps d'exécution très différents.

7.3.2.3 Mode *shell* distant

Le deuxième mode de parallélisation consiste à utiliser les propriétés des *shells* distants tels que *rsh* ou *ssh* pour créer des processus distants et communiquer avec ces derniers. Dans ce mode de communication, les processus de calcul lisent sur l'entrée standard les informations dont ils ont besoin et écrivent sur la sortie standard leurs résultats. Dans le cas des *shells rsh* et *ssh*, l'entrée standard de la commande *shell* est propagée à l'entrée standard du processus de calcul sur la machine distante et la sortie standard du processus distant est propagée sur la sortie standard de la commande *shell* (cf. figure 7.3). Il est alors naturel d'utiliser l'entrée et la sortie standards de la commande shell comme moyen de communication avec le processus.

Afin de paralléliser au mieux l'ensemble des tâches par le mode *shell* tout en prenant en compte les contraintes liées à ce mode de communication, la fonction *Run* de l'objet *Parallelizer* est défini par l'algorithme 7.2.

Algorithme 7.1: *Parallelizer : :Run* en mode parallélisation MPI

Tant que (des tâches sont encore en cours d'exécution) **et** (il reste des tâches à paralléliser) **Faire**
  Tant que (il reste des tâches à paralléliser) **Faire**
    Allouer un processus sur une machine
    Si l'allocation échoue **Alors**
      Terminer la boucle **Tant que**
    Sinon
      Démarrer une tâche sur le processus alloué
    Fin Si
  Fin Tant que
  Si (il n'y a pas de tâches en exécution) **et**
     (il reste des tâches à paralléliser) **Alors**
    Exécuter une tâche localement
  Fin Si
  Si il y a des tâches en exécution **Alors**
    Attendre la fin d'une tâche
    Terminer la tâche
    Libérer le processus de la machine
  Fin Si
Fin Tant que

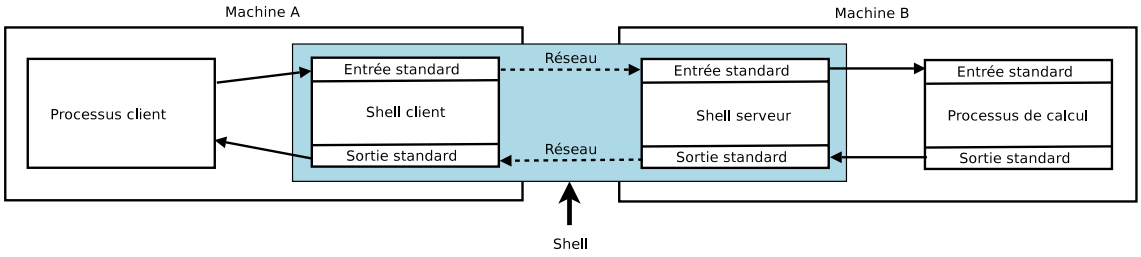


FIG. 7.3 – Propagation de l'entrée standard et de la sortie standard à l'aide d'un shell distant.

Tout comme précédemment, le lecteur remarquera que, pour ne pas tester (test non bloquant) un grand nombre de fois si une tâche est terminée, dans le cas où il reste des tâches à exécuter, nous avons choisi d'exécuter une tâche localement afin de profiter du temps processeur disponible pour le processus client.

Dans l'implémentation actuelle de la bibliothèque, il est possible d'utiliser n'importe quel *shell* distant. En effet, la commande *shell* utilisée pour la création des processus distants est définie dans les structures de données du *daemon* de gestion des ressources. Ces commandes sont fournies au *daemon* au lancement de celui-ci. Dans nos expérimentations, nous avons exploité les *shells* distants *rsh* (NetKit, 2005) et *ssh* (OpenSSH, 2005).

7.3.2.4 Mode *socket* client-server

L'inconvénient des deux modes de communication précédents est qu'ils nécessitent la création dynamique des processus de calcul, ce qui peut être très coûteux en temps processeur si ce nombre de créations est important par rapport au temps de calcul de chacun des processus. Une solution alternative, que nous avons considérée, est un mode de fonctionnement en client-server : des processus serveurs de calculs sont créés avant et les clients contactent ces serveurs pour leur demander de faire certains calculs. En raison du principe de décomposition

des tâches en sous-tâches, un processus serveur peut également se retrouver dans la position d'un client demandeur envers un autre processus. Les communications s'effectuent via des *sockets*.

Algorithme 7.2: *Parallelizer : :Run* en mode parallélisation *shell* distant

```
Tant que (des tâches sont encore en cours d'exécution) et (il reste des tâches à paralléliser) Faire
    Tant que (il reste des tâches à paralléliser) Faire
        Allouer un processus sur une machine
        Si l'allocation échoue Alors
            Terminer la boucle Tant que
        Sinon
            Démarrer une tâche sur le processus alloué
        Fin Si
    Fin Tant que
    Si il y a des tâches en exécution Alors
        Si une tâche est en attente de terminaison Alors
            Terminer la tâche
            Libérer le processus de la machine
        Sinon
            Si il reste des tâches à exécuter Alors
                Exécuter une tâche localement
            Fin Si
        Fin Si
    Fin Si
Fin Tant que
```

Afin de paralléliser au mieux l'ensemble des tâches par le mode *shell* tout en prenant en compte les contraintes liées à ce mode de communication, la fonction *Run* de l'objet *Parallelizer* est défini par l'algorithme 7.3.

Tout comme précédemment, le lecteur remarquera que, pour ne pas tester (test non bloquant) un grand nombre de fois si une tâche est terminée, dans le cas où il reste des tâches à exécuter, nous avons choisi d'exécuter une tâche localement, afin de profiter du temps processeur disponible pour le processus client.

7.4 Parallélisation des métaheuristiques pour l'apprentissage de MMC

La parallélisation des métaheuristiques à base de population, telles que les algorithmes génétiques, les fourmis artificielles ou l'optimisation par essaim particulaire consistent la plupart du temps à paralléliser le déplacement de chacun des agents. Cette approche, bien que simple, n'est pas bien adaptée au cas de l'apprentissage de MMC. En effet, cette approche nécessite le transfert d'un trop grand nombre de données par rapport aux calculs effectués. Or, on peut remarquer que ce qui est le plus coûteux en temps de calcul, c'est l'optimisation locale d'une solution. Nous proposons donc de paralléliser les calculs de manière à ne transferer au maximum que les données d'un modèle. Les métaheuristiques doivent donc être restructurées.

La parallélisation efficace des algorithmes génétiques pour l'apprentissage de MMC consiste simplement à effectuer les calculs d'optimisation locale en parallèle.

Algorithme 7.3: *Parallelizer : :Run* en mode parallélisation *socket* client-serveur

Tant que (des tâches sont encore en cours d'exécution) **et** (il reste des tâches à paralléliser) **Faire**
  Tant que (il reste des tâches à paralléliser) **Faire**
    Allouer un processus sur une machine
    Si l'allocation échoue **Alors**
      Terminer la boucle **Tant que**
    Sinon
      Démarrer une tâche sur le processus alloué
    Fin Si
  Fin Tant que
  Si il y a des tâches en exécution **Alors**
    Si une tâche est en attente de terminaison **Alors**
      Terminer la tâche
      Libérer le processus de la machine
    Sinon
      Si il reste des tâches à exécuter **Alors**
        Exécuter une tâche localement
      Fin Si
    Fin Si
  Fin Si
Fin Tant que

La parallélisation efficace de l'algorithme de fourmis artificielles API s'effectue en décomposant les déplacements des fourmis en trois étapes : la détermination du site à explorer, l'exploration du site et la mise à jour de la mémoire de la fourmi. En considérant cette décomposition, il est possible de paralléliser la deuxième étape. L'algorithme obtenu est donné par l'algorithme 7.4.

Algorithme 7.4: La métaheuristique API parallèle

$s_{\text{Nid}} = \mathcal{O}_{\text{Rand}}(\mathcal{S})$
$\mathcal{M}(a_i) = \oslash$ pour tout $i = 1..M$
Pour $t = 1$ **à** $\mathcal{T}_{\max}$ **Faire**
  Déterminer les sites à explorer pour chacune des fourmis par l'algorithme 7.5
  Exploration en parallèle des sites avec les opérateurs $\mathcal{O}_{\text{ExploSite}}$ et $\mathcal{O}_{\text{ExploLocal}}$
  Mise à jour des fourmis par l'algorithme 7.6
  Si t modulo $\mathcal{T}_{\text{Déplacement}} = 0$ **Alors**
    $s_{\text{Nid}} = \max\{s_{\text{Nid}}, \max_{i=1..M} s^*(a_i)\}$
    $\mathcal{M}(a_i) = \emptyset$ pour tout $i = 1..M$
  Fin Si
Fin Pour
$s^* = \max\{s_{\text{Nid}}, \max_{i=1..M} s^*(a_i)\}$

La parallélisation efficace de l'algorithme d'optimisation par essaim particulaire pour l'apprentissage de MMC suit les mêmes principes. Le mouvement des particules est décomposé en deux morceaux : le déplacement de la particule et la mise à jour de la vitesse. Cette première étape incluant l'ajout du vecteur vitesse et l'optimisation locale est effectuée en parallèle pour toutes les particules. La deuxième étape, de coût négligeable par rapport au coût qu'elle aurait si on la parallélisait, est exécuté en séquence sur les particules.

Algorithme 7.5: Détermination du site à explorer et du type d'exploration d'une fourmi

Si $|\mathcal{M}| < \mathcal{M}_{\max}$ **Alors**
  $s = s_{\text{Nid}}$
  Exploration de type $\mathcal{O}_{\text{ExploSite}}$
Sinon
  Si *echec* **Alors**
    $last = \mathcal{U}(\mathcal{M})$
    $s = last$
    Exploration de type $\mathcal{O}_{\text{ExploLocal}}(last)$
  Fin Si
Fin Si

Comme nous pouvons le constater, la parallélisation de ces algorithmes peut s'effectuer simplement et plus efficacement que l'approche naïve. Cette organisation parallèle a pour avantage de ne pas augmenter la complexité des algorithmes si la parallélisation n'est pas activée dans la bibliothèque.

7.5 Les modules

Comme nous l'avons défini dans les objectifs, la bibliothèque est composée de plusieurs modules pouvant être utilisés séparément. Les différents modules disponibles actuellement sont :

- *Core* : gestion des exceptions, structures de données (vecteur, matrice, ...), flux, flux de communication, gestion des références sur les objets, architecture de parallélisation, *Parallelizer*, *daemon* de gestion des ressources
- *MetaTK* : ensemble de classes patrons (*i.e.* des *templates*) définissant les principales métaheuristiques utilisées par la bibliothèque
- *DiscretHMM* : implémentation des algorithmes *Forward*, *Backward*, Baum-Welch, Viterbi, AG, API, PSO, ... pour les modèles de Markov cachés
- SSHMM : implémentation des algorithmes *Forward*, *Backward*, Baum-Welch, Viterbi, Viterbi « étendu », AG, API, PSO, ... pour les modèles de Markov cachés à substitutions de symboles
- MDIHMM : implémentation des algorithmes *Forward*, *Backward*, Baum-Welch, Viterbi, AG, API, PSO, ... pour les modèles de Markov cachés multidimensionnels à processus indépendants (Brouard, 1999).

Les modules DiscretHMM, SSHMM et MDIHMM possèdent une structuration identique : la première partie correspond à la définition et à l'implémentation des fonctions parallélisables, tandis que la deuxième partie utilise les fonctions définies dans la première partie afin de constituer le processus de calcul associé. Par conséquent, chaque module construit son processus de calcul, ce qui signifie que le *daemon* de gestion des ressources doit être informé de la façon de créer chacun de ces processus sur une machine distante. Cette approche peut paraître complexe au premier abord, mais elle permet une totale modularité et extensibilité de la bibliothèque.

Afin de réduire la quantité de code et par là même améliorer la fiabilité de la bibliothèque, chaque module peut nécessiter la présence d'un ou plusieurs modules pour fonctionner :

- *Core* : ne nécessite aucun autre module, mais il nécessite une bibliothèque MPI2, telle que LAM/MPI, dans le cas d'une parallélisation par MPI
- *MetaTK* : ne nécessite aucun module, car il ne s'agit que d'un ensemble d'entêtes et de classes patrons

- *DiscretHMM* : nécessite *Core* et *MetaTK*
- *SSHMM* : nécessite *Core*, *MetaTK* et *DiscretHMM*
- *MDIHMM* : nécessite *Core*, *MetaTK*

Algorithme 7.6: Mise à jour d'une fourmi

Si $|\mathcal{M}| < \mathcal{M}_{\max}$ **Alors**
  Si $(\mathcal{M} = \emptyset)$ **ou** $(f(s^*) < f(s))$ **Alors**
    $s^* = s$
  Fin Si
  $\mathcal{M} = \mathcal{M} \cup \{s\}$
  $e(s) = 0$
  $echec = Vrai$
Sinon
  Si *echec* **Alors**
    Si $f(s) > f(last)$ **Alors**
      $\mathcal{M} = \mathcal{M} - \{last\}$
      $\mathcal{M} = \mathcal{M} \cup \{s\}$
      $e(s) = 0$
      $echec = Faux$
      Si $f(s^*) < f(s)$ **Alors**
        $s^* = s$
      Fin Si
    Sinon
      $echec = Vrai$
      $e(last) = e(last) + 1$
      Si $e(last) = e_{\max}$ **Alors**
        $\mathcal{M} = \mathcal{M} - \{last\}$
      Fin Si
    Fin Si
  Fin Si
Fin Si

7.6 Conclusion

La bibliothèque HMMTK, que nous avons développée, remplit entièrement les objectifs que nous nous sommes fixés au départ : généralité, modularité, efficacité, extensibilité et facilité de parallélisation. La bibliothèque sera d'ici peu rendue publique sur le site `http://www.hant.li.univ-tours.fr` probablement sous licence CeCILL v2 (CeCILL, 2005). En l'état actuel, la bibliothèque permet la manipulation de trois types de modèles de Markov cachés : les MMC, les MMCSS et les MMC multidimensionnels à processus indépendants. A terme, nous prévoyons d'implémenter d'autres types de modèles de Markov cachés, afin d'enrichir la bibliothèque. Finalement, nous prévoyons plusieurs améliorations en terme de temps de calcul et de collaboration avec d'autres langages, tels que le C, le Java (Java, 2005), le Python (Python, 2005) et le Perl (Perl, 2005).

Conclusions et perspectives

Ce qu'il faut retenir

Les modèles de Markov cachés (MMC) (cf. chapitre 1) sont des outils statistiques permettant de modéliser de nombreux phénomènes stochastiques. Ils sont utilisés dans de nombreux domaines avec efficacité. La vive concurrence provenant d'autres outils, tels que les réseaux de neurones ou les machines à supports de vecteurs, rend nécessaire d'améliorer les performances des MMC. Pour cela, nous avons envisagé trois directions de recherche.

La première direction que nous avons empruntée concerne l'apprentissage des modèles. Cet apprentissage peut être vu comme un problème d'optimisation consistant à trouver les paramètres du modèle qui maximisent un certain critère. De nombreux critères peuvent être utilisés (cf. chapitre 1). Pour chacun des critères présentés, nous avons montré qu'il existait un algorithme permettant de trouver un modèle localement optimal du critère. Pour obtenir un MMC qui soit un optimum global du critère, il est nécessaire d'employer des techniques permettant de le rechercher. De nombreuses métaheuristiques ont été développées (cf. chapitre 2), cependant la plupart n'ont été que sommairement décrites et leurs performances ont été relativement peu comparées entre elles. La non disponibilité du code de ces algorithmes empêche de mener des comparatifs poussés. Par conséquent, plusieurs méthodes sont disponibles, mais on ne sait pas si l'une d'entre elles est meilleure que les autres. Partant de ce constat, nous avons conçu de nouveaux algorithmes d'apprentissage et nous les avons comparés à deux algorithmes existants (cf. chapitre 4). Nous nous sommes intéressés aux espaces de solutions. Nous avons défini trois espaces de solutions : convexe/borné, discret et vectoriel. Ces trois espaces nous ont permis d'envisager de compléter les adaptations de trois métaheuristiques génériques (algorithme génétique et fourmis artificielles API, optimisation par essaim particulaire) sur ces espaces. Nous avons donc adapté l'algorithme génétique à l'espace discret, l'algorithme API aux espaces « discrets et vectoriels » et l'optimisation par essaim particulaire à l'espace vectoriel. Après avoir comparé les métaheuristiques par rapport à leur stratégie d'exploration, nous avons déterminé des configurations de «bons» paramètres pour chacun de ces algorithmes, puis nous avons étudié les performances de ces algorithmes.

La deuxième direction (cf. chapitre 5) que nous avons empruntée concerne l'amélioration des capacités des MMC, sans pour autant augmenter drastiquement la complexité des algorithmes classiques. Pour cela, nous avons défini les modèles de Markov cachés à substitutions de symboles (MMCSS). Ces modèles permettent l'incorporation de connaissances expertes dans le modèle sous la forme d'une loi de probabilité modélisant la substitution d'un « méta » symbole par un symbole dans la séquence observée. Après avoir redéfini les algorithmes classiques (*Forward*, *Backward*, Baum-Welch, Gradient, Viterbi, Viterbi « étendu »), nous avons étudié les capacités de ces nouveaux modèles pour la classification et la segmentation d'images.

La troisième direction que nous avons explorée consiste à expliquer l'effet d'un ensemble de MMC sur un jeu de données grâce à la visualisation d'informations (cf. chapitre 3). Pour cela, nous avons mis en relation des MMC à l'aide d'une dissimilarité. L'utilisation d'une dissimilarité permet d'étudier les aspects que l'on veut tout en gardant une grande flexibilité dans la définition de ses valeurs. Nous avons alors cherché à visualiser le nuage de points

induit par la dissimilarité (cf. chapitre 6). A cet effet, nous avons cherché à étendre l'analyse en composantes principales à noyau (ACPN) au cas de noyaux indéfinis. Nous avons montré que le plongement du nuage dans un espace pseudo-euclidien permettait de le faire et que, pour une base bien choisie, la forme quadratique associée est une somme de carrés. À partir de ce plongement et des propriétés des espaces pseudo-euclidiens obtenus, nous avons adapté la technique de la matrice de *scatterplots* de façon à permettre la visualisation et l'interprétation du nuage. Cette nouvelle technique est dénommée matrice de *scatterplots* pseudo-euclidienne (MSPE). Finalement, nous avons montré sur un exemple comment la MSPE peut être utilisée pour comparer des dissimilarités sur des MMC et mieux comprendre ce que modélisent les MMC.

Parallèlement à l'ensemble de ces travaux, nous avons développé une bibliothèque de modèles de Markov cachés dénommée HMMTK. Cette bibliothèque adopte un paradigme de programmation objet dans lequel les données, comme les fonctions, sont des objets. Ce paradigme a permis d'inclure dans la bibliothèque un ensemble de mécanismes de parallélisation des calculs (exécution séquentielle, parallélisation par MPI, parallélisation par *shell* distant, parallélisation client/serveur) de manière quasi transparente pour l'utilisateur. Nous avons également restructuré les métaheuristiques d'algorithmes génétiques, de fourmis artificielles API et d'optimisation par essaim particulaire, afin de produire les mêmes résultats en mode séquentiel qu'en mode parallèle, tout en évitant le transfert en trop grosse quantité de données sur le réseau.

Perspectives

Les travaux commencés au cours de cette thèse ouvrent la voie à de nombreux travaux futurs.

En effet, concernant le premier axe de recherche *i.e.* l'apprentissage de MMC, il reste plusieurs directions à explorer. L'adaptation des algorithmes génétiques à l'espace de solutions vectoriel ou l'adaptation de l'optimisation par essaim particulaire à l'espace de solutions discret en sont deux exemples. De même, il serait intéressant d'évaluer et de comparer les métaheuristiques du chapitre 2 aussi bien sur l'espace de solutions convexe/borné que sur les espaces de solutions discret et vectoriel. Dans la même optique, il serait probablement intéressant de construire un algorithme d'apprentissage de MMC utilisant dans un premier temps l'espace de solutions discret puis dans un deuxième temps l'espace de solutions convexe/borné ou vectoriel. Il serait également intéressant de comparer ces métaheuristiques aux heuristiques telles que le *Stochastic* EM ou SAEM. Les expérimentations, que nous avons menées, ont été réalisées pour le critère de maximum de vraisemblance. Elles nous ont permis de mettre en évidence un ensemble de « bons » paramètres pour chacun des algorithmes. Ces paramètres sont suffisamment robustes pour que les performances des algorithmes ne changent pas d'une séquence d'observations à une autre. Il serait donc intéressant de vérifier si ces configurations de « bons » paramètres restent viables pour d'autres critères d'apprentissage, et dans le cas contraire, s'il est possible de déterminer de « bons » paramètres robustes pour les métaheuristiques.

Les MMCSS promettent également de nombreux travaux. En effet, nos expérimentations ont permis de montrer l'intérêt des MMCSS pour la classification et la segmentation d'images. La possibilité d'inclure à moindre coût des connaissances expertes dans l'apprentissage et la reconnaissance ouvre la voie à de nouvelles applications, telles que le traitement des documents textuels (indexation automatique, classification et reconnaissance automatique (Serradura et al., 2001), ...).

La structure pseudo-euclidienne définie au chapitre 6 offre également de nombreuses possibilités. En effet, avec les connaissances acquises sur ces espaces, il nous est d'ores et déjà possible d'envisager d'autres techniques de visualisation de dissimilarité. L'apport des noyaux indéfinis pour la visualisation de dissimilarité conforte l'idée émise dans (Pekalska et al., 2002) et (Ong et al., 2004) comme quoi les méthodes à noyaux ne sont pas forcément tributaires de noyaux semi-définis positifs. Par conséquent, il nous paraît intéressant, dans un futur proche, d'explorer les capacités de ces techniques pour la classification de données à l'aide d'une dissimilarité sur des MMC.

Finalement, le développement de la bibliothèque HMMTK n'en est qu'à ses débuts. Nous prévoyons de la développer de manière importante afin d'en faire un outil classique. Au titre des améliorations, nous envisageons d'interfacer la bibliothèque avec d'autres langages tels que le C, le Java, le Python et le Perl. Nous envisageons d'inclure de nouveaux modèles (MMC avec mélange gaussien, ...) et d'améliorer les performances de la parallélisation, notamment en réduisant les temps de latence dans les communications.

Troisième partie

Annexes

Annexe A

Démonstration de l'algorithme de Baum-Welch

Dans le cas des MMC, on cherche à maximiser $P(V = O/\lambda)$ avec O une séquence de T observations. En appliquant l'algorithme EM à la maximisation de cette probabilité (Bilmes, 1998), on est amené à maximiser $\Gamma(\lambda, \lambda')$ avec $\lambda = (A, B, \Pi)$ le nouveau modèle et λ' le modèle connu (ou actuel) :

$$\Gamma_O(\lambda, \lambda') = \sum_{Q \in \mathbb{S}^T} P(S = Q/V = O, \lambda') \ln P(V = O, S = Q/\lambda)$$

Sachant que

$$P(V = O, S = Q/\lambda) = \pi_{q_1} \left(\prod_{t=1}^{T-1} a_{q_t q_{t+1}} \right) \left(\prod_{t=1}^{T} b_{q_t}(o_t) \right)$$

la fonction Γ_O se ré-écrit

$$\begin{aligned}
\Gamma_O(\lambda, \lambda') &= \sum_{Q \in \mathbb{S}^T} \ln \pi_{q_1} P(S = Q/V = O, \lambda') \\
&+ \sum_{Q \in \mathbb{S}^T} \left(\sum_{t=1}^{T-1} \ln a_{q_t q_{t+1}} \right) P(S = Q/V = O, \lambda') \\
&+ \sum_{Q \in \mathbb{S}^T} \left(\sum_{t=1}^{T} \ln b_{q_t}(o_t) \right) P(S = Q/V = O, \lambda') \\
&= \Gamma_O^{\pi}(\lambda, \lambda') + \Gamma_O^{A}(\lambda, \lambda') + \Gamma_O^{B}(\lambda, \lambda')
\end{aligned}$$

On peut alors remarquer que $\Gamma(\lambda, \lambda')$ se décompose en la somme de trois fonctions de paramètres distincts et indépendants, par conséquent il est possible de les maximiser indépendamment les uns des autres.

A.1 Ré-estimation des π_i

On peut dans un premier temps remarquer que le premier côté de l'égalité n'impose que le premier état caché donc on a

$$\begin{aligned}
\Gamma_O^{\pi}(\lambda, \lambda') &= \sum_{Q \in \mathbb{S}^T} \ln \pi_{q_1} P(S = Q/V = O, \lambda') \\
&= \sum_{i=1}^{N} \ln \pi_i P(S_1 = s_i/V = O, \lambda')
\end{aligned}$$

En utilisant les multiplicateurs de Lagrange pour contraindre $\sum_{i=1}^{N} \pi_i = 1$ et en dérivant, on obtient (Bilmes, 1998)

$$\begin{aligned} \frac{\partial}{\partial \pi_i}\left(\Gamma_O^{\pi}(\lambda,\lambda') + \gamma\left(\sum_{i=1}^{N}\pi_i - 1\right)\right) &= \frac{P(S_1 = s_i/V = O,\lambda')}{\pi_i} + \gamma \\ &= 0 \end{aligned}$$

et

$$\begin{aligned} \frac{\partial}{\partial \gamma}\left(\Gamma_O^{\pi}(\lambda,\lambda') + \gamma\left(\sum_{i=1}^{N}\pi_i - 1\right)\right) &= \sum_{i=1}^{N}\pi_i - 1 \\ &= 0 \end{aligned}$$

En multipliant la première équation par π_i et en sommant sur $i = 1..N$, on obtient grâce à la deuxième équation $\gamma = -1$ et donc (Bilmes, 1998)

$$\pi_i = P(S_1 = s_i/O,\lambda')$$

A.2 Ré-estimation des a_{ij}

Tout comme précédemment, il est possible de simplifier l'expression car :

$$\begin{aligned} \Gamma_O^{A}(\lambda,\lambda') &= \sum_{Q\in\mathbb{S}^T}\left(\sum_{t=1}^{T-1}\ln a_{q_t,q_{t+1}}\right)P(S = Q/V = O,\lambda') \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{t=1}^{T-1}\ln a_{ij}P(S_t = s_i, S_{t+1} = s_j/V = O,\lambda') \end{aligned}$$

car la première formule revient à fixer les états cachés en t et $t+1$.

En utilisant les multiplicateurs de Lagrange pour contraindre $\sum\limits_{j=1}^{N} a_{ij} = 1$ et en dérivant, on obtient (Bilmes, 1998)

$$\begin{aligned} &\frac{\partial}{\partial a_{i,j}}\left(\Gamma_O^{A}(\lambda,\lambda') + \sum_{n=1}^{N}\gamma_n\left(\sum_{k=1}^{N}a_{n,k} - 1\right)\right) \\ &= \sum_{t=1}^{T-1}\frac{P(S_t = s_i, S_{t+1} = s_j/V = O,\lambda')}{a_{i,j}} + \gamma_i \\ &= 0 \end{aligned}$$

et

$$\begin{aligned} \frac{\partial}{\partial \gamma_i}\left(\Gamma_O^{A}(\lambda,\lambda') + \sum_{n=1}^{N}\gamma_n\left(\sum_{k=1}^{N}a_{n,k} - 1\right)\right) &= \sum_{k=1}^{N}a_{i,k} - 1 \\ &= 0 \end{aligned}$$

En multipliant par $a_{i,j}$ et en sommant sur j la première équation, on obtient grâce à la deuxième équation :

$$\gamma_i = -\sum_{t=1}^{T-1}P(S_t = s_i/V = O,\lambda')$$

et donc

$$a_{i,j} = \frac{\sum_{t=1}^{T-1} P(S_t = s_i, S_{t+1} = s_j / V = O, \lambda')}{\sum_{t=1}^{T-1} P(S_t = s_i / V = O, \lambda')}$$

A.3 Ré-estimation des $b_i(j)$

Pour des raisons similaires aux précédentes, il est possible de simplifier l'expression (Bilmes, 1998) :

$$\begin{aligned} \Gamma_O^B(\lambda, \lambda') &= \sum_{Q \in \mathbb{S}^T} \left(\sum_{t=1}^{T} \ln b_{q_t}(o_t) \right) P(S = Q / V = O, \lambda') \\ &= \sum_{i=1}^{N} \sum_{t=1}^{T} \ln b_i(o_t) P(S_t = s_i / V = O, \lambda') \end{aligned}$$

En utilisant les multiplicateurs de Lagrange pour contraindre $\sum_{j=1}^{M} b_i(j) = 1$ et en dérivant, on obtient

$$\begin{aligned} &\frac{\partial}{\partial b_i(j)} \left(\Gamma_O^B(\lambda, \lambda') + \sum_{n=1}^{N} \gamma_n \left(\sum_{j=1}^{N} b_n(j) - 1 \right) \right) \\ &= \sum_{t=1}^{T} \frac{P(S_t = s_i / V = O, \lambda') \delta(o_t = j)}{b_i(j)} + \gamma_i \\ &= 0 \end{aligned}$$

et

$$\begin{aligned} \frac{\partial}{\partial \gamma_i} \left(\Gamma_O^B(\lambda, \lambda') + \sum_{n=1}^{N} \gamma_n \left(\sum_{j=1}^{N} b_n(j) - 1 \right) \right) &= \sum_{j=1}^{N} b_i(j) - 1 \\ &= 0 \end{aligned}$$

La fonction $\delta(p)$ vaut 1 si le prédicat p est vrai et 0 sinon. On remarque alors que $\sum_{j=1}^{M} \delta(o_t = j) = 1$, car un et un seul symbole o_t peut se réaliser à un temps donné.

Alors en multipliant par $b_i(j)$ la première équation et en sommant sur j, on obtient, grâce à la deuxième équation

$$\gamma_i = -\sum_{t=1}^{T} P(q_t = i / V = O, \lambda')$$

et donc

$$b_i(j) = \frac{\sum_{t=1}^{T} P(S_t = s_i / V = O, \lambda') \delta(o_t = j)}{\sum_{t=1}^{T} P(S_t = s_i / V = O, \lambda')}$$

A.4 Synthèse

On obtient :

$$\begin{aligned}
\pi_i &= P(S_1 = s_i/O, \lambda') \\
a_{i,j} &= \frac{\sum_{t=1}^{T-1} P(S_t = s_i, S_{t+1} = s_j/V = O, \lambda')}{\sum_{t=1}^{T-1} P(S_t = s_i/V = O, \lambda')} \\
b_i(j) &= \frac{\sum_{t=1}^{T} P(S_t = s_i/V = O, \lambda')\delta(o_t = j)}{\sum_{t=1}^{T} P(S_t = s_i/V = O, \lambda')}
\end{aligned}$$

Annexe B

Démonstration du calcul du gradient de la vraisemblance

Soit $L(\lambda) = \ln P(V = O/\lambda)$. Maximiser la vraisemblance $P(V = O/\lambda)$ est équivalent à maximiser le logarithme de la vraisemblance $L(\lambda)$. Calculons le gradient de $L(\lambda)$ par rapport aux paramètres de λ au point μ.

$$\frac{\partial L(\lambda)}{\partial \lambda}(\mu) = \frac{1}{P(V = O/\mu)} \frac{\partial P(V = O/\lambda)}{\partial \lambda}(\mu)$$

Maximiser $L(\lambda)$ nécessite donc de calculer les dérivées partielles de $P(V = O/\lambda)$ par rapport aux différents paramètres du modèle. Afin de ne pas gérer les contraintes de stochasticité et établir un gradient réutilisable, nous exprimons le modèle avec les variables $x_{i,j}$, $y_{i,j}$ et z_i.

B.1 Dérivées partielles des matrices stochastiques

On note $\kappa(cond) = 1$ si $cond$ est vrai et $\kappa(cond) = 0$ sinon. On montre facilement que

$$\begin{aligned} \frac{\partial a_{i,l}}{\partial x_{i,j}} &= a_{i,l}(\kappa(l = j) - a_{i,j}) \\ \frac{\partial b_i(m)}{\partial y_{i,j}} &= b_i(m)(\kappa(m = j) - b_i(j)) \\ \frac{\partial \pi_l}{\partial z_i} &= \pi_l(\kappa(i = l) - \pi_i) \end{aligned}$$

B.2 Dérivées partielles par rapport aux variables $x_{i,j}$

En utilisant la règle de chaînage pour la dérivation de $g(\theta) = f(h_1(\theta), \ldots, h_k(\theta))$

$$\frac{\partial g}{\partial \theta} = \sum_{i=1}^{k} \frac{\partial f}{\partial h_i} \cdot \frac{\partial h_i}{\partial \theta}$$

on obtient

$$\frac{\partial P(V = O/\lambda)}{\partial x_{i,j}} = \sum_{t=1}^{T} \sum_{l=1}^{N} \frac{\partial P(V = O/\lambda)}{\partial \alpha_t(l)} \cdot \frac{\partial \alpha_t(l)}{\partial a_{i,l}} \cdot \frac{\partial a_{i,l}}{\partial x_{i,j}}$$

or

$$\alpha_t(l) = \begin{cases} \pi_l b_l(o_1) & \text{si } t = 1 \\ \sum_{k=1}^{N} \alpha_{t-1}(k) a_{k,l} b_l(o_t) & \text{si } t > 1 \end{cases}$$

donc

$$\frac{\partial \alpha_t(l)}{\partial a_{i,l}} = \begin{cases} 0 & \text{si } t = 1 \\ \alpha_{t-1}(i) b_l(o_t) & \text{si } t > 1 \end{cases}$$

or $P(V = O/\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i)$ donc

$$\frac{\partial P(V = O/\lambda)}{\partial \alpha_t(l)} = \beta_t(l)$$

d'où

$$\begin{aligned} \frac{\partial P(V = O/\lambda)}{\partial x_{i,j}} &= \sum_{t=2}^{T} \sum_{l=1}^{N} \beta_t(l)\alpha_{t-1}(i) b_l(o_t) a_{i,l}(\kappa(l = j) - a_{i,j}) \\ &= \sum_{t=2}^{T} \alpha_{t-1}(i) a_{i,j} b_j(o_t)\beta_t(j) - a_{i,j} \sum_{t=2}^{T} \alpha_{t-1}(i)\beta_{t-1}(i) \end{aligned}$$

B.3 Dérivées partielles par rapport aux variables $y_{i,j}$

De façon similaire,

$$\begin{aligned} \frac{\partial P(V = O/\lambda)}{\partial y_{i,j}} &= \sum_{t=1}^{T} \sum_{l=1}^{N} \sum_{m=1}^{N} \frac{\partial P(V = O/\lambda)}{\partial \alpha_t(l)} \cdot \frac{\partial \alpha_t(l)}{\partial b_l(o_t = m)} \cdot \frac{\partial b_l(o_t = m)}{\partial y_{i,j}} \\ &= \sum_{t=1}^{T} \sum_{m=1}^{N} \frac{\partial P(V = O/\lambda)}{\partial \alpha_t(i)} \cdot \frac{\partial \alpha_t(i)}{\partial b_i(o_t = m)} \cdot \frac{\partial b_i(o_t = m)}{\partial y_{i,j}} \\ &= \sum_{t=1}^{T} \sum_{m=1}^{N} \frac{\partial P(V = O/\lambda)}{\partial \alpha_t(i)} \cdot \frac{\partial \alpha_t(i)}{\partial b_i(m)} \cdot \frac{\partial b_i(m)}{\partial y_{i,j}} \end{aligned}$$

$$\alpha_t(i) = \begin{cases} \pi_i b_i(o_1) & \text{si } t = 1 \\ \sum_{k=1}^{N} \alpha_{t-1}(k) a_{k,i} b_i(o_t) & \text{si } t > 1 \end{cases}$$

donc

$$\frac{\partial \alpha_t(i)}{\partial b_i(m)} = \begin{cases} \kappa(o_1 = m)\pi_i & \text{si } t = 1 \\ \kappa(o_t = m) \sum_{k=1}^{N} \alpha_{t-1}(j) a_{k,i} & \text{si } t > 1 \end{cases}$$

or

$$\kappa(o_t = m) \left\{ \begin{array}{ll} \pi_i b_i(m) & \text{si } t = 1 \\ \left(\sum_{k=1}^{N} \alpha_{t-1}(k) a_{k,i} \right) b_i(m) & \text{si } t > 1 \end{array} \right\} = \kappa(o_t = m)\alpha_t(i)$$

d'où

$$\begin{aligned} \frac{\partial P(V = O/\lambda)}{\partial y_{i,j}} &= \sum_{t=1}^{T} \sum_{m=1}^{N} \beta_t(i)\kappa(o_t = m)\alpha_t(i)(\kappa(m = j) - b_i(j)) \\ &= \sum_{t=1}^{T} \beta_t(i)\alpha_t(i)(\kappa(o_t = j) - b_i(j)) \end{aligned}$$

B.4 Dérivées partielles par rapport aux variables z_i

De façon similaire,

$$\frac{\partial P(V=O/\lambda)}{\partial z_i} = \sum_{l=1}^{N} \frac{\partial P(V=O/\lambda)}{\partial \pi_l} . \frac{\partial \pi_l}{\partial z_i}$$

or $P(V=O/\lambda) = \sum_{k=1}^{N} \pi_i b_i(o_1)\beta_1(i)$ donc

$$\frac{\partial \pi_l}{\partial z_i} = b_l(o_1)\beta_1(l)$$

d'où

$$\begin{aligned} \frac{\partial P(V=O/\lambda)}{\partial z_i} &= \sum_{l=1}^{N} \pi_l b_l(o_1)\beta_1(l)(\kappa(i=l) - \pi_i) \\ &= \pi_i b_i(o_1)\beta_1(i) - \pi_i P(V=O/\lambda) \end{aligned}$$

B.5 Synthèse

On obtient :

$$\begin{aligned} \frac{\partial P(V=O/\lambda)}{\partial x_{i,j}} &= \sum_{t=2}^{T} \alpha_{t-1}(i) a_{i,j} b_j(o_t)\beta_t(j) - a_{i,j} \sum_{t=2}^{T} \alpha_{t-1}(i)\beta_{t-1}(i) \\ \frac{\partial P(V=O/\lambda)}{\partial y_{i,j}} &= \sum_{t=1}^{T} \beta_t(i)\alpha_t(i)(\kappa(o_t=j) - b_i(j)) \\ \frac{\partial P(V=O/\lambda)}{\partial z_i} &= \pi_i b_i(o_1)\beta_1(i) - \pi_i P(V=O/\lambda) \end{aligned}$$

Annexe C

La métaheuristique API

Dans cette annexe, nous détaillons, de manière plus complète qu'au chapitre 2, la métaheuristique API. Cette métaheuristique est complètement définie par les algorithmes C.1 et C.2.

Algorithme C.1: La métaheuristique API

// Notations :
// $\mathcal{M}(a_i)$: la mémoire de la fourmi a_i
// $s^*(a_i)$: la meilleure solution rencontrée par la fourmi a_i
$s_{\text{Nid}} = \mathcal{O}_{\text{Rand}}(\mathcal{S})$
$\mathcal{M}(a_i) = \oslash$ pour tout $i = 1..M$
Pour $t = 1$ **à** $\mathcal{T}_{\max}$ **Faire**
  Pour $i = 1$ **à** $\mathcal{N}$ **Faire**
    Exécuter l'algorithme C.2 sur la fourmi a_i
  Fin Pour
  Si t modulo $\mathcal{T}_{\text{Déplacement}} = 0$ **Alors**
    $s_{\text{Nid}} = \max\{s_{\text{Nid}}, \max_{i=1..M} s^*(a_i)\}$
    $\mathcal{M}(a_i) = \emptyset$ pour tout $i = 1..M$
  Fin Si
Fin Pour
$s^* = \max\{s_{\text{Nid}}, \max_{i=1..M} s^*(a_i)\}$

Algorithme C.2: Fourragement de la fourmi a_i dans l'algorithme API

// $\mathcal{M}$: la mémoire de la fourmi
// $\mathcal{M}_{\max}$: la taille maximale de la mémoire
// $last$: la dernière solution explorée
// $e(s)$: le nombre d'échecs mémorisés lors de l'exploration de la solution s
// $echec$: est-ce que la dernière exploration est un échec ?
// s^* : la meilleure solution rencontrée par la fourmi
Si $|\mathcal{M}| < \mathcal{M}_{\max}$ **Alors**
  $s = \mathcal{O}_{\text{ExploSite}}(s_{\text{Nid}})$
  Si $(\mathcal{M} = \emptyset)$ **ou** $(f(s^*) < f(s))$ **Alors**
    $s^* = s$
  Fin Si
  $\mathcal{M} = \mathcal{M} \cup \{s\}$
  $e(s) = 0$
  $echec = Vrai$
Sinon
  Si $echec$ **Alors**
    $last = \mathcal{U}(\mathcal{M})$
  Fin Si
  $s = \mathcal{O}_{\text{ExploLocal}}(last)$
  Si $f(s) > f(last)$ **Alors**
    $\mathcal{M} = \mathcal{M} - \{last\}$
    $\mathcal{M} = \mathcal{M} \cup \{s\}$
    $e(s) = 0$
    $echec = Faux$
    Si $f(s^*) < f(s)$ **Alors**
      $s^* = s$
    Fin Si
  Sinon
    $echec = Vrai$
    $e(last) = e(last) + 1$
    Si $e(last) = e_{\max}$ **Alors**
      $\mathcal{M} = \mathcal{M} - \{last\}$
    Fin Si
  Fin Si
Fin Si

Annexe D

Graphes du chapitre 4 « Métaheuristiques pour l'apprentissage de MMC »

Cette annexe regroupe une partie des graphes obtenus au chapitre 4 afin de faciliter la lecture du document.

D.1 Comparaison des performances

Les graphiques suivants présentent les courbes moyennes de la probabilité du meilleur modèle trouvé en fonction du nombre de modèles évalués (*i.e.* du nombre de *Forward* lorsque l'algorithme de Baum-Welch n'est pas utilisé) et du nombre total d'itérations de l'algorithme de Baum-Welch effectuées lorsque l'algorithme de Baum-Welch est utilisé.

Pour les tests, nous avons considéré les configurations de « bons » paramètres obtenues précédemment. Ces configurations sont :

- AG1 : *MuterParents*=Non, $p_{\text{mut}} = 0.01$, $\mathcal{N} = 5$
- AG2 : $\mathcal{N} = 5$, $p_{\text{mut}} = 0.3$, *MuterParents*=Oui et *OptimiserParents*=Oui
- AG3 : $\mathcal{N} = 20$, $p_{\text{mut}} = 0.3$, *MuterParents*=Non et *OptimiserParents*=Oui
- AG4 : $\mathcal{N} = 20$, $p_{\text{mut}} = 0.5$, *MuterParents*=Oui et *OptimiserParents*=Oui
- AG5 : $\mathcal{N} = 5$, $p_{\text{mut}} = 0.5$, *MuterParents*=Non et *OptimiserParents*=Oui
- APIHomo1 : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 50$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.1$, $e_{\max} = 1$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHomo2 : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 5$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.1$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHomo3 : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 5$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.1$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHete1 : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 2$, $e_{\max} = 5$, $\mathcal{T}_{\text{Déplacement}} = 10$
- APIHete2 : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 5$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHete3 : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 5$, $e_{\max} = 3$, $\mathcal{T}_{\text{Déplacement}} = 5$
- OEPDistance1 : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 100$, $\omega = 1$, $c_1 = 0.5$, $c_2 = 0.5$, $d = d_0$
- OEPDistance2 : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 20$, $\omega = 0.4$, $c_1 = 1$, $c_2 = 0$
- OEPDistance3 : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 50$, $\omega = 0.2$, $c_1 = 0$, $c_2 = 0$, $V = 6$
- OEPSocial1 : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 20$, $\omega = 0.8$, $c_1 = 1.5$, $c_2 = 2.5$, $V = 9$
- OEPSocial2 : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 20$, $\omega = 0.8$, $c_1 = 1.5$, $c_2 = 0$
- OEPSocial3 : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 50$, $\omega = 0.4$, $c_1 = 0$, $c_2 = 0.5$, $V = 6$
- AGDiscret *MuterParents*=Non, $p_{\text{mut}} = 0.01$, $\mathcal{N} = 5$
- APIDiscretHomo : $e_{\max} = 1$, $\mathcal{N} = 2$, $\mathcal{T}_{\text{Déplacement}} = 5$, $\mathcal{A}^i_{\text{site}} = 0.1$ et $\mathcal{A}^i_{\text{local}} = 0.2$
- APIDiscretHete : $e_{\max} = 1$, $\mathcal{N} = 2$, $\mathcal{T}_{\text{Déplacement}} = 5$
- APIHomo1* : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 2$, $\mathcal{A}^i_{\text{local}} = 0.9$, $\mathcal{A}^i_{\text{site}} = 0.8$, $e_{\max} = 1$, $\mathcal{T}_{\text{Déplacement}} = 5$
- APIHomo2* : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 20$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.2$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 5$

- APIHomo3* : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 50$, $\mathcal{A}^i_{\text{local}} = 0.1$, $\mathcal{A}^i_{\text{site}} = 0.2$, $e_{\max} = 4$, $\mathcal{T}_{\text{Déplacement}} = 15$
- APIHete1* : $\mathcal{N}_{\text{BW}} = 0$, $\mathcal{N} = 2$, $e_{\max} = 5$, $\mathcal{T}_{\text{Déplacement}} = 4$
- APIHete2* : $\mathcal{N}_{\text{BW}} = 2$, $\mathcal{N} = 20$, $e_{\max} = 5$, $\mathcal{T}_{\text{Déplacement}} = 2$
- APIHete3* : $\mathcal{N}_{\text{BW}} = 5$, $\mathcal{N} = 50$, $e_{\max} = 20$, $\mathcal{T}_{\text{Déplacement}} = 1$

Les algorithmes Random0, Random2 et Random5 correspondent à une recherche alétoire dans Λ. Pour chacune des solutions envisagées, 0, 2 ou 5 itérations de Baum-Welch sont appliquées.

Les séquences d'observations considérées pour ces comparaisons sont obtenues à partir des quatre images utilisées précédemment mais en faisant varier :

- le nombre de symboles lors du ré-échantillonnement *i.e.* M,
- le nombre d'états cachés dans les modèles *i.e.* N.

D.1.1 Lorsque $M = 32$, $N = 11$ et $T = 400$

Les figures D.1, D.2, D.3 et D.4 donnent les graphiques de performance lorsque $M = 32$, $N = 11$ et $T = 400$.

D.1.2 Lorsque $M = 64$, $N = 11$ et $T = 400$

Les figures D.5, D.6, D.7 et D.8 donnent les graphiques de performance lorsque $M = 64$, $N = 11$ et $T = 400$.

D.1.3 Lorsque $M = 32$, $N = 20$ et $T = 400$

Les figures D.9, D.10, D.11 et D.12 donnent les graphiques de performance lorsque $M = 32$, $N = 20$ et $T = 400$.

D.2 Comparaison des meilleurs algorithmes

Les figures D.13, D.14, D.15 et D.16 présentent les graphes de la moyenne du logarithme de la vraisemblance obtenus pour les configurations AG2, AG4, APIHete2, APIHete3, Random2, Random5, Random10, Random100 et Random1000.

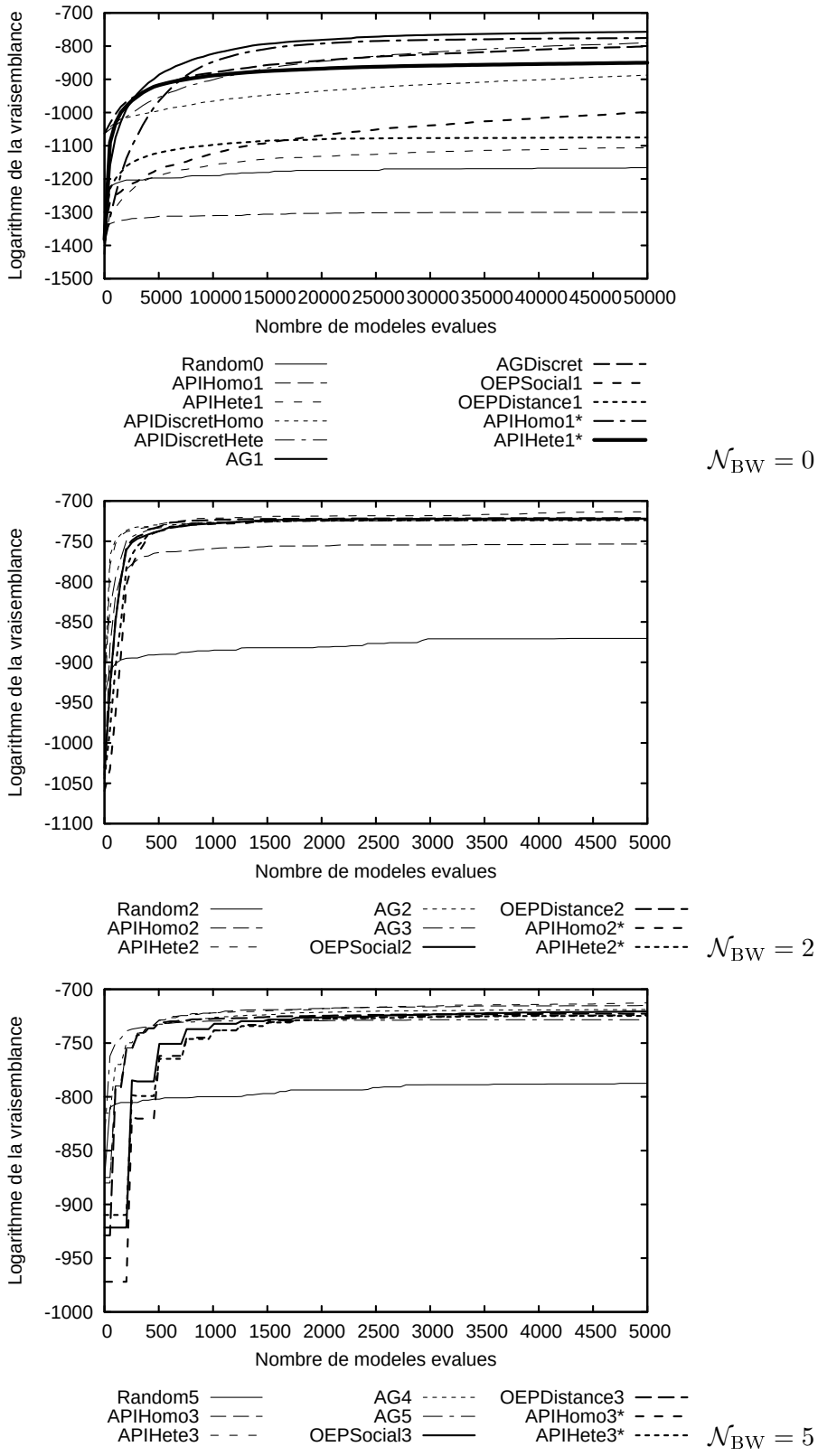


FIG. D.1 – Moyenne du logarithme de la vraisemblance pour $Img1$, $M = 32$, $N = 11$, $T = 400$.

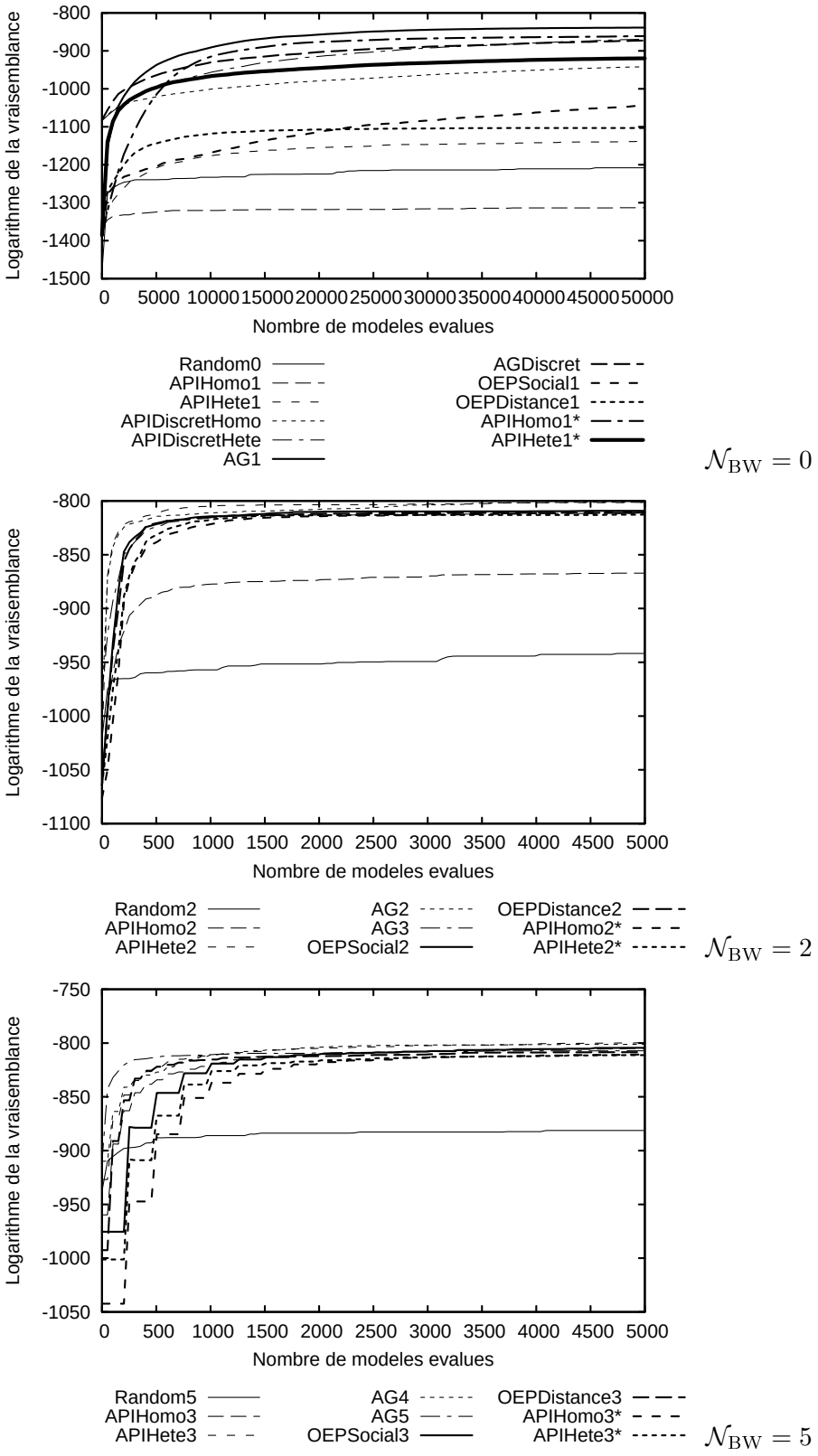


FIG. D.2 – Moyenne du logarithme de la vraisemblance pour $Img2$, $M = 32$, $N = 11$, $T = 400$.

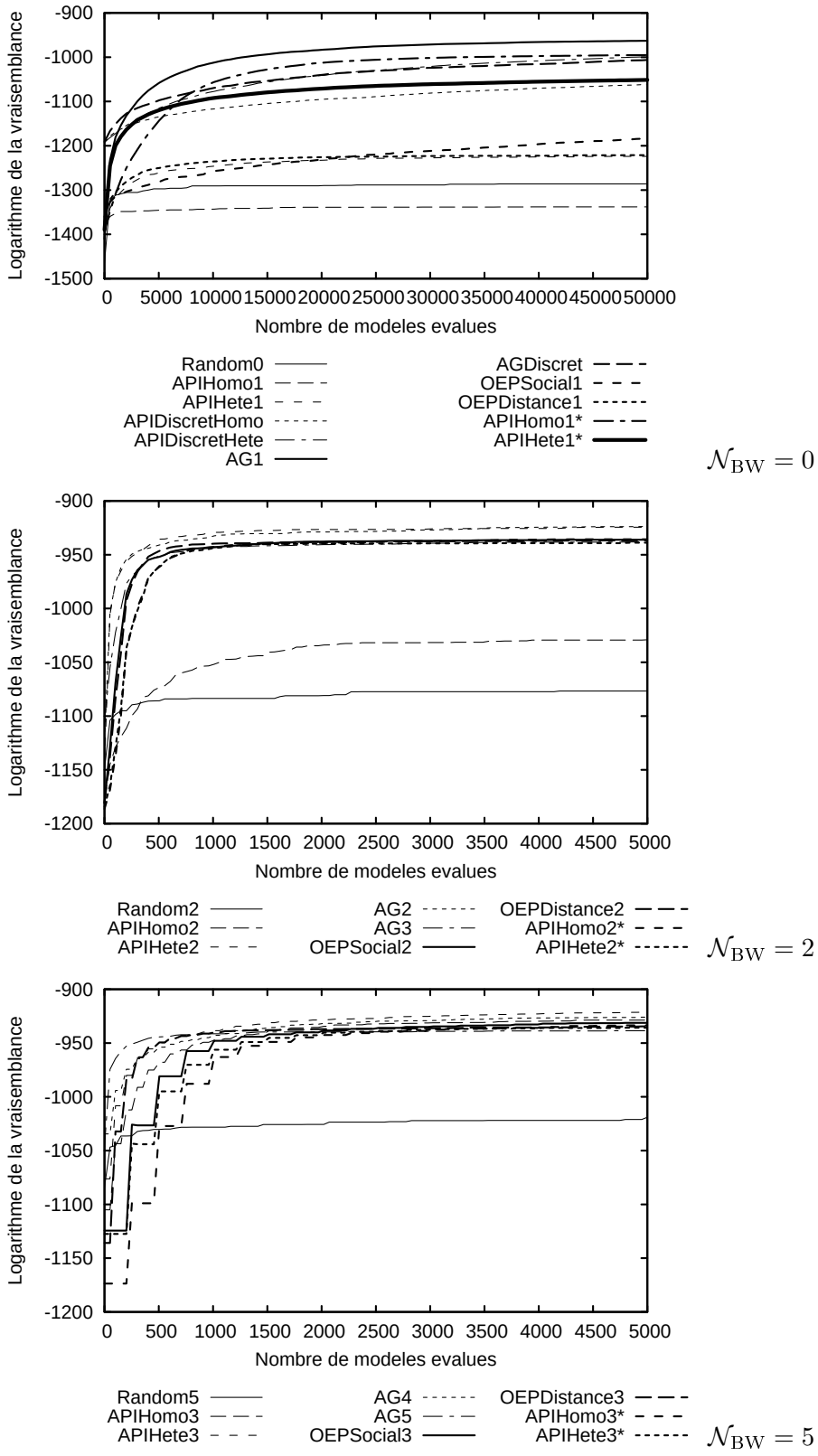
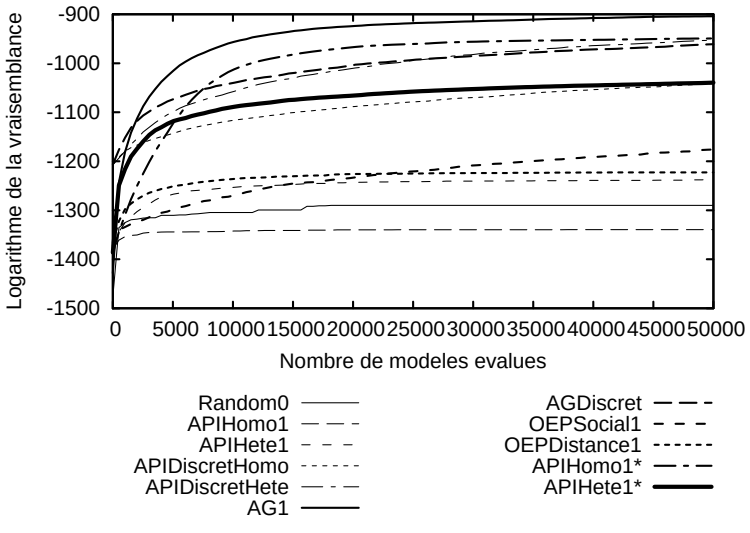
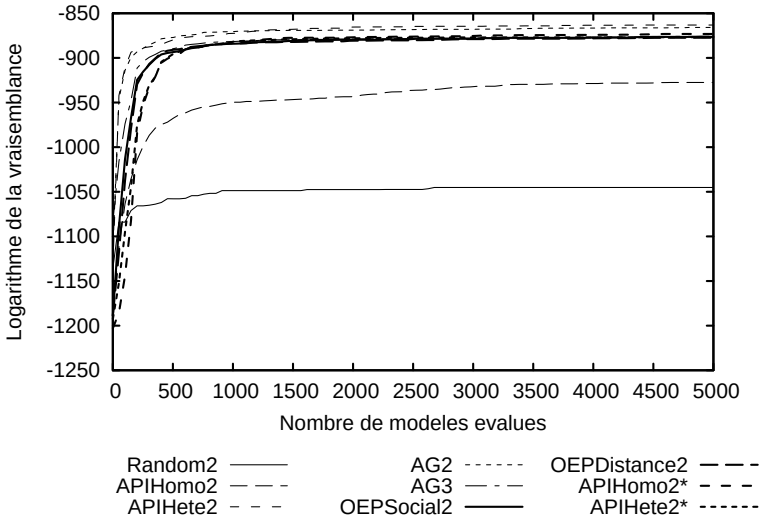


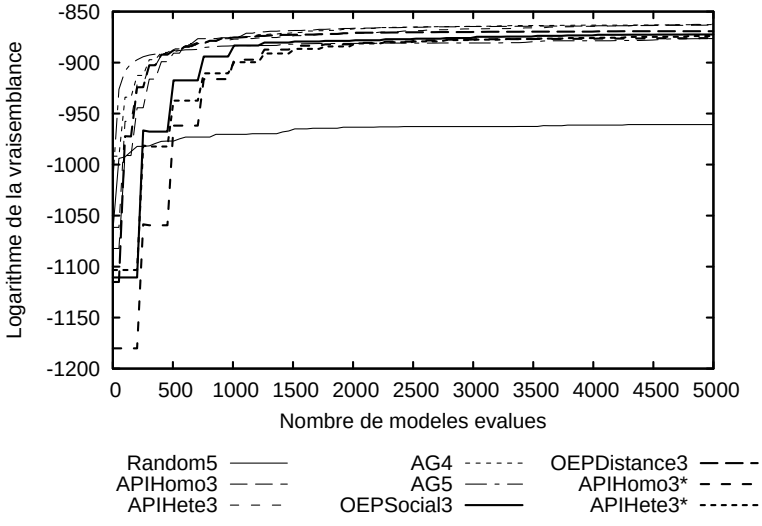
FIG. D.3 – Moyenne du logarithme de la vraisemblance pour $Img3$, $M = 32$, $N = 11$, $T = 400$.



$\mathcal{N}_{\mathrm{BW}} = 0$



$\mathcal{N}_{\mathrm{BW}} = 2$



$\mathcal{N}_{\mathrm{BW}} = 5$

FIG. D.4 – Moyenne du logarithme de la vraisemblance pour $Img4$, $M = 32$, $N = 11$, $T = 400$.

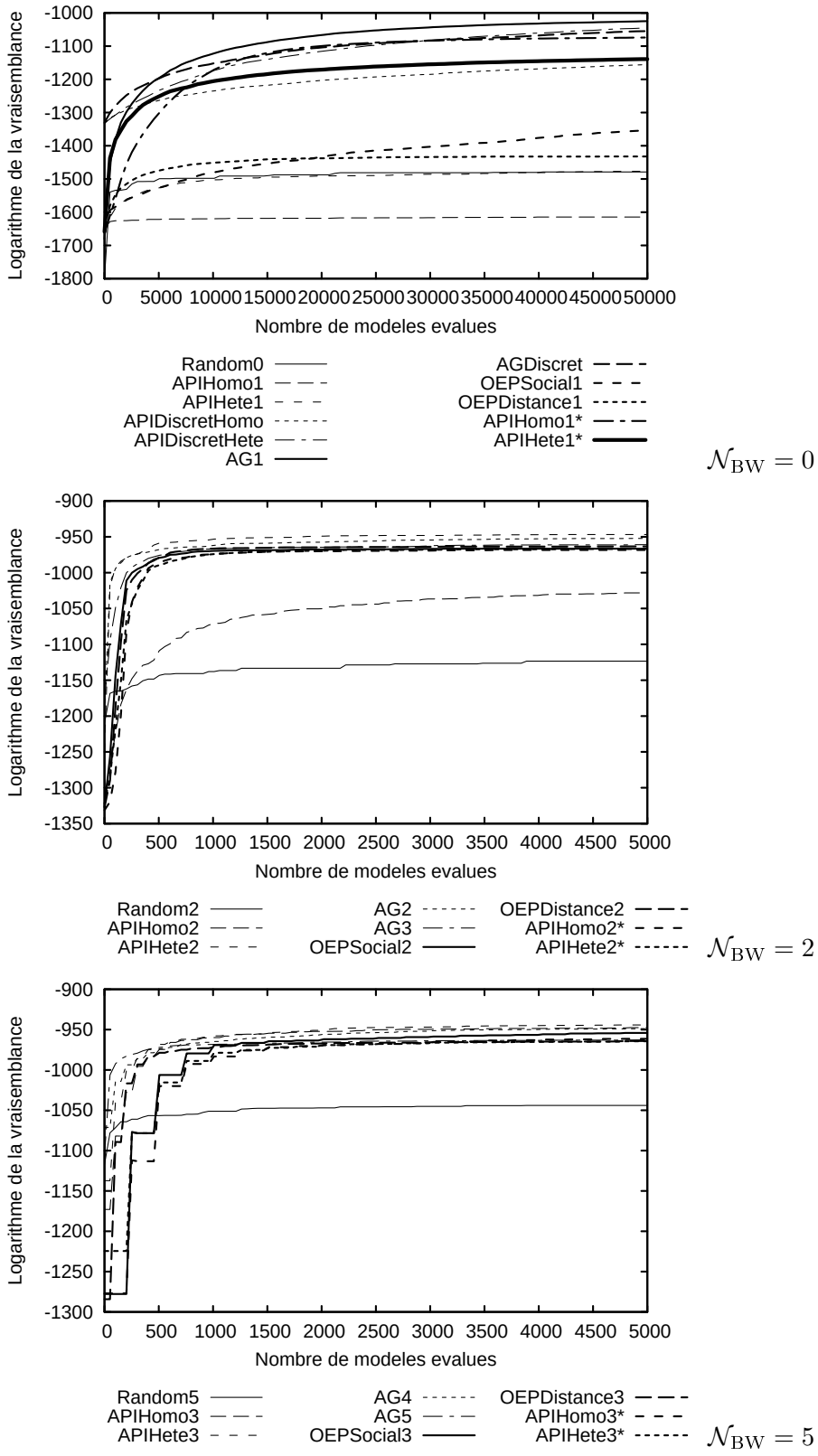
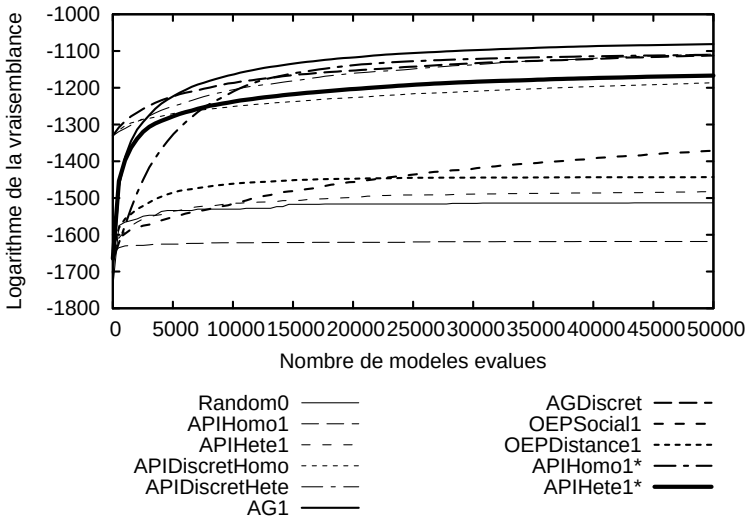
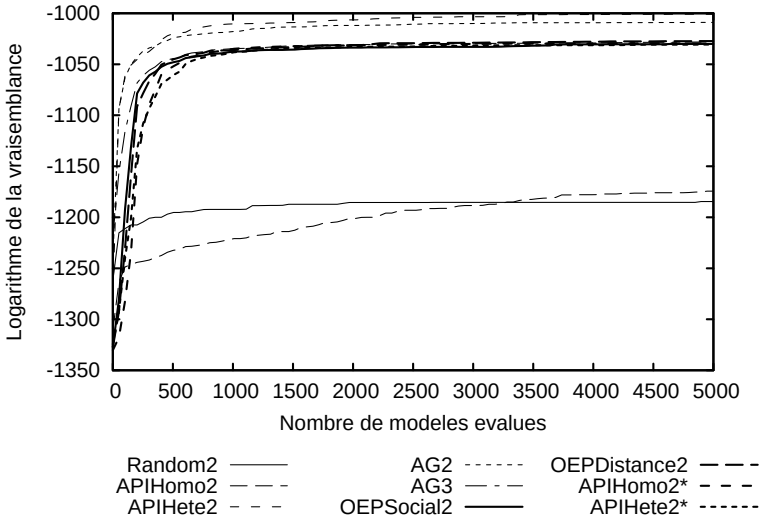


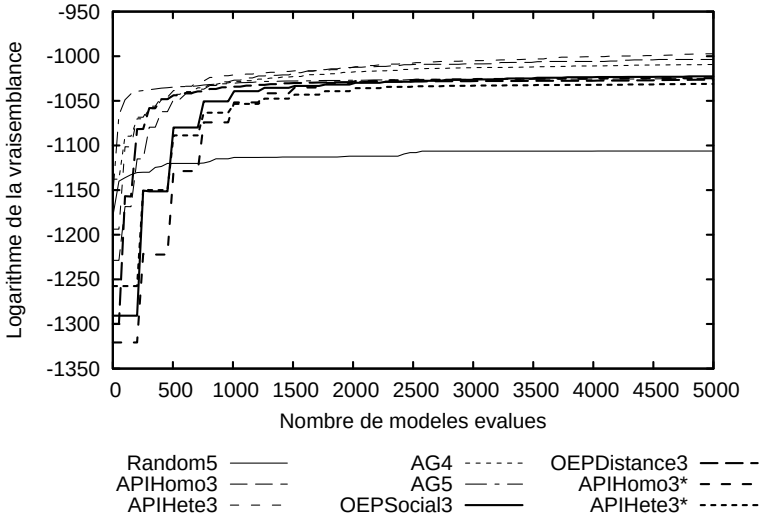
FIG. D.5 – Moyenne du logarithme de la vraisemblance pour $Img1$, $M = 64$, $N = 11$, $T = 400$.



$\mathcal{N}_{\mathrm{BW}} = 0$



$\mathcal{N}_{\mathrm{BW}} = 2$



$\mathcal{N}_{\mathrm{BW}} = 5$

FIG. D.6 – Moyenne du logarithme de la vraisemblance pour $Img2$, $M = 64$, $N = 11$, $T = 400$.

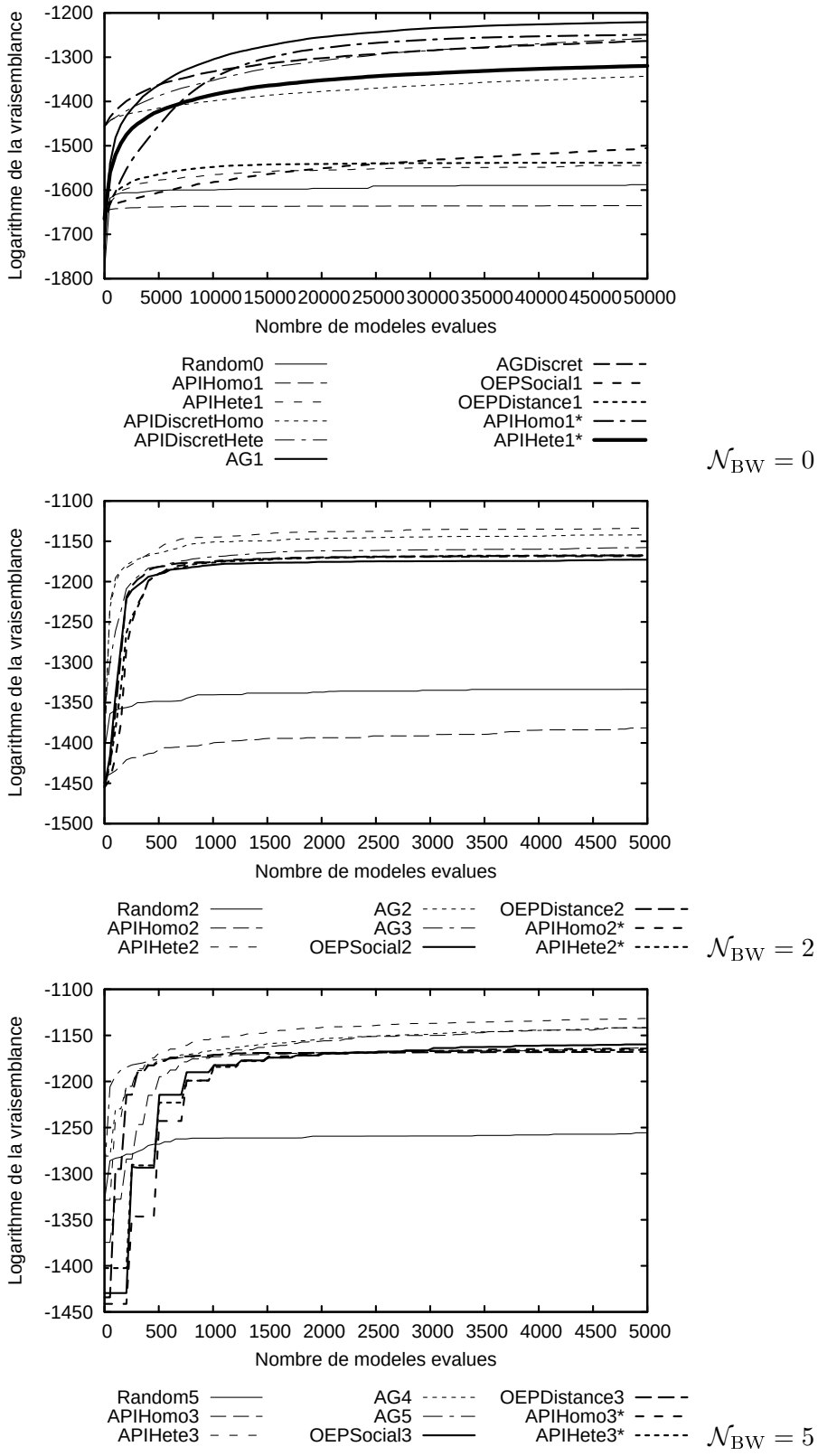
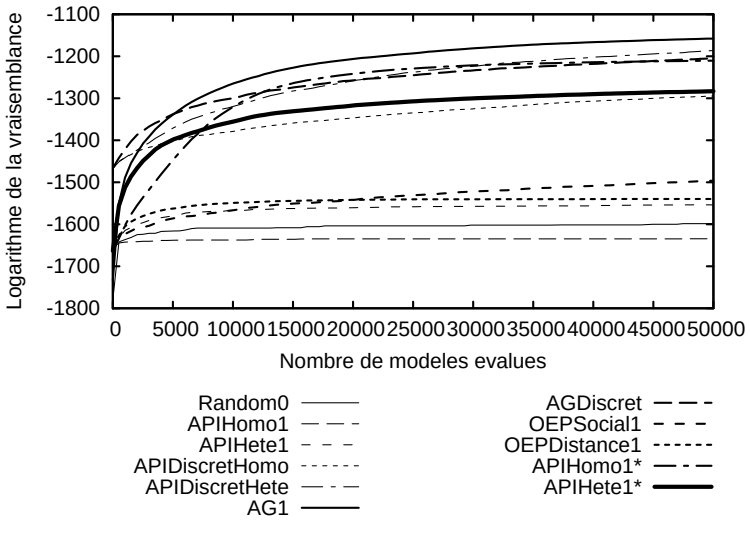
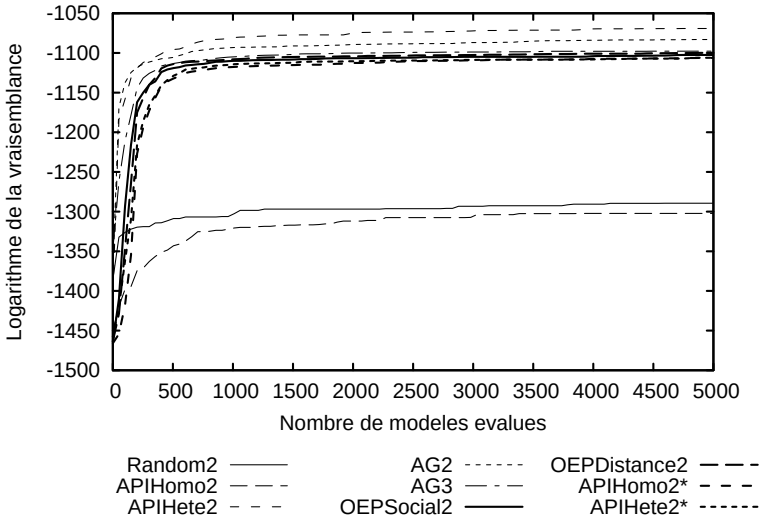


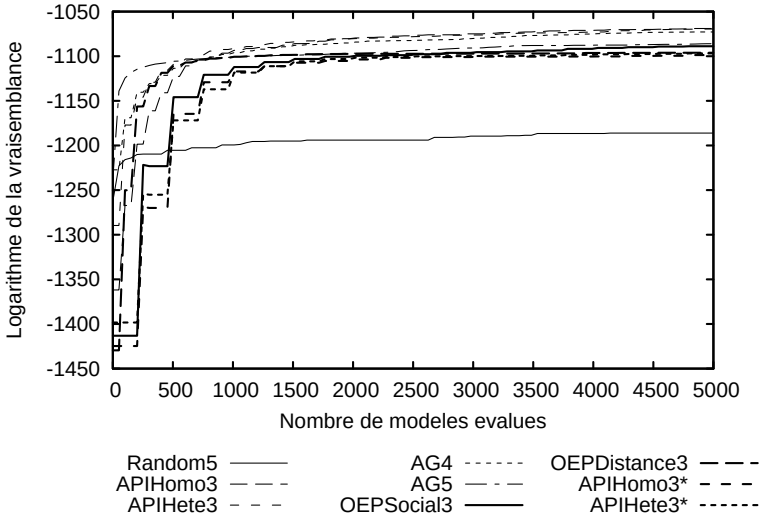
FIG. D.7 – Moyenne du logarithme de la vraisemblance pour $Img3$, $M = 64$, $N = 11$, $T = 400$.



$\mathcal{N}_{\mathrm{BW}} = 0$



$\mathcal{N}_{\mathrm{BW}} = 2$



$\mathcal{N}_{\mathrm{BW}} = 5$

FIG. D.8 – Moyenne du logarithme de la vraisemblance pour $Img4$, $M = 64$, $N = 11$, $T = 400$.

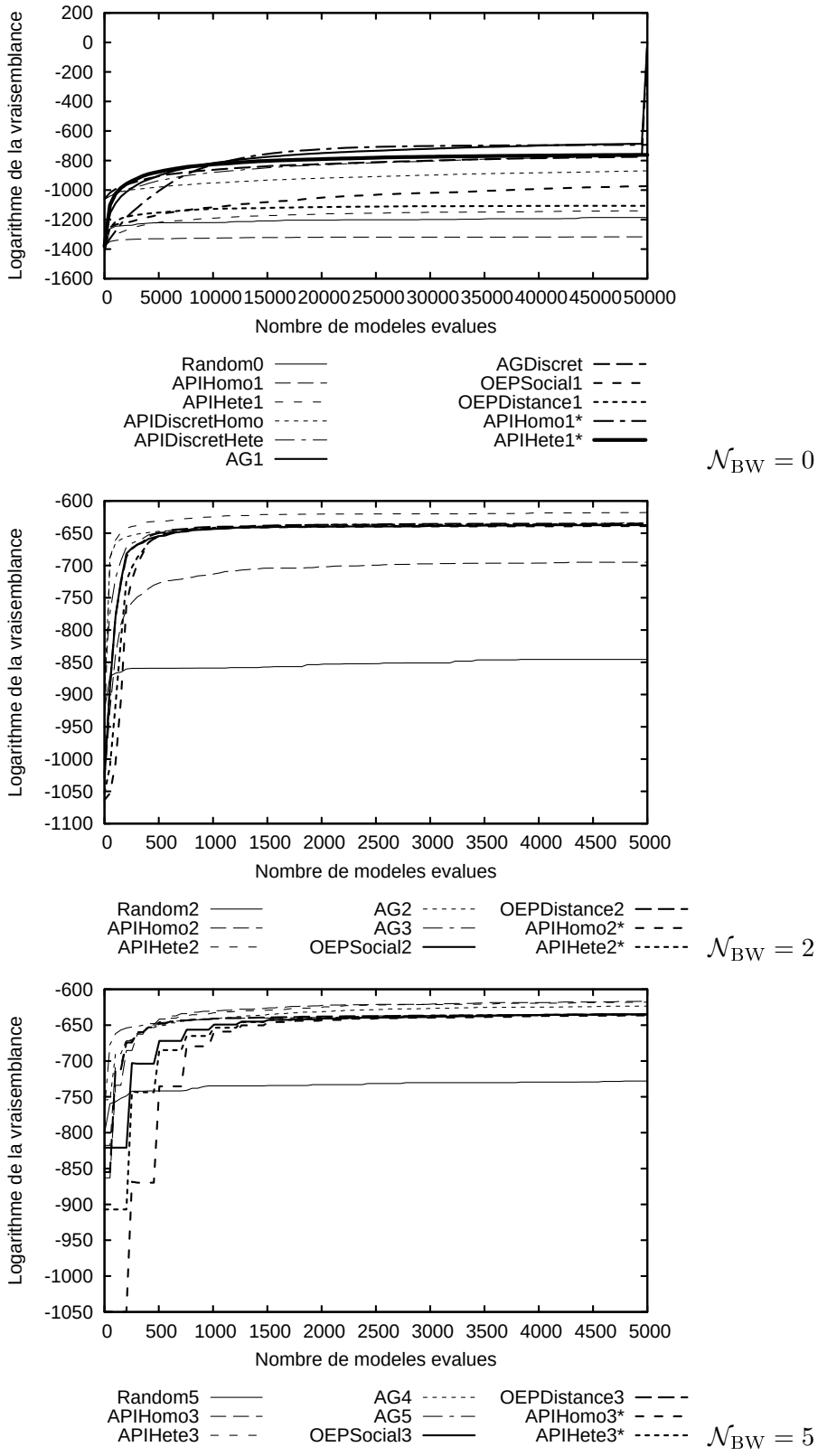


FIG. D.9 – Moyenne du logarithme de la vraisemblance pour $Img1$, $M = 32$, $N = 20$, $T = 400$.

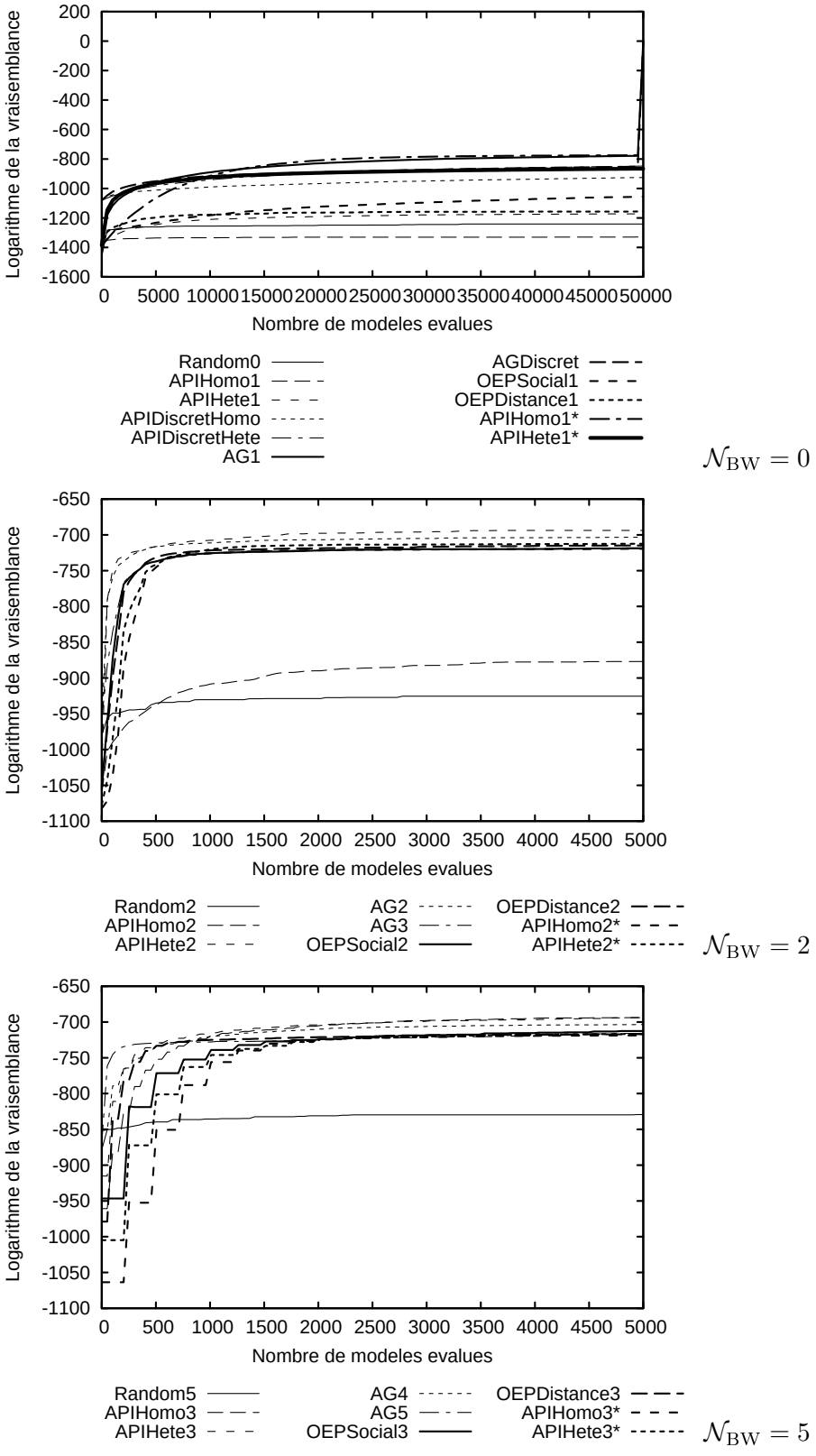


FIG. D.10 – Moyenne du logarithme de la vraisemblance pour $Img2$, $M = 32$, $N = 20$, $T = 400$.

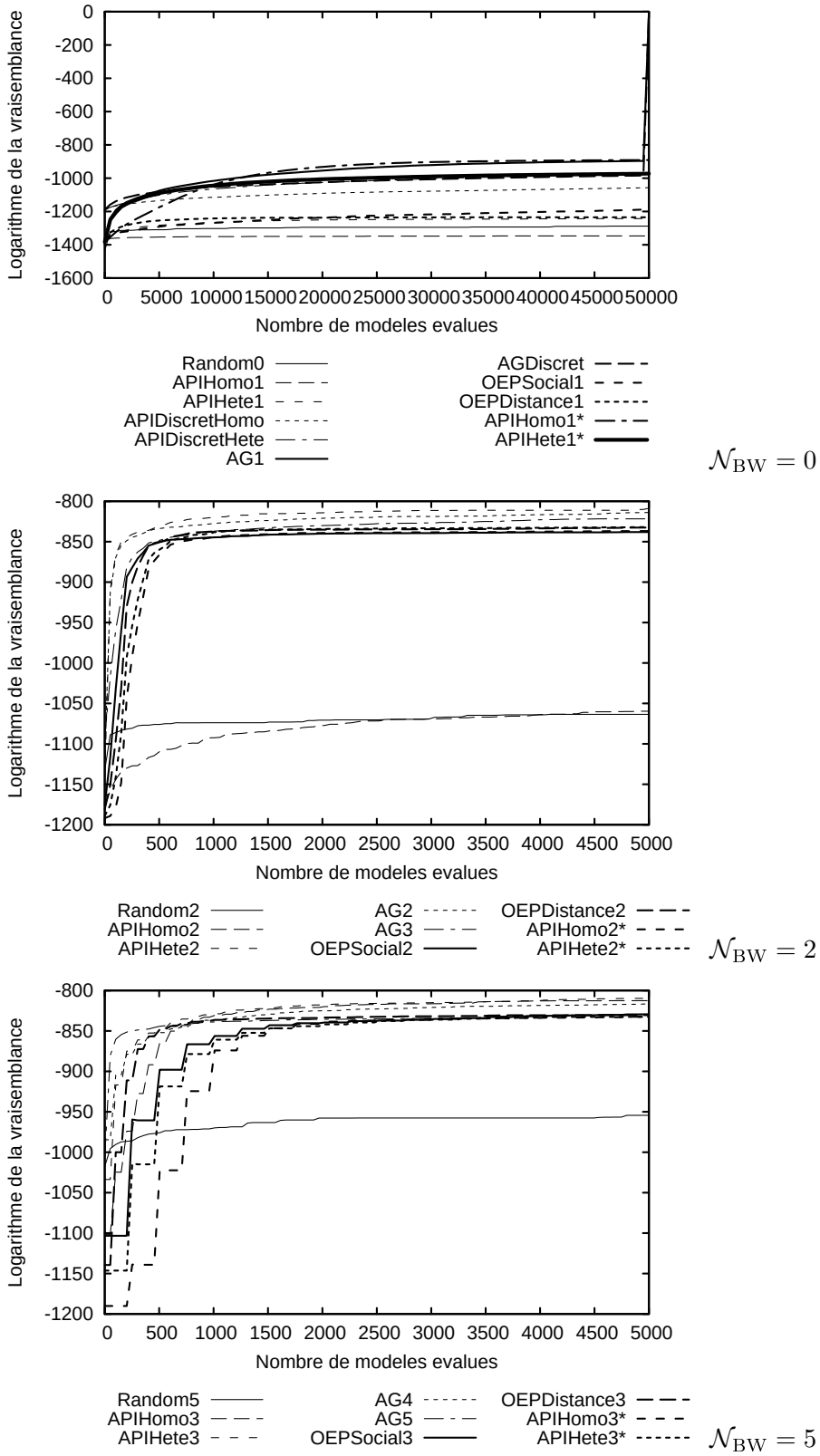


FIG. D.11 – Moyenne du logarithme de la vraisemblance pour $Img3$, $M = 32$, $N = 20$, $T = 400$.

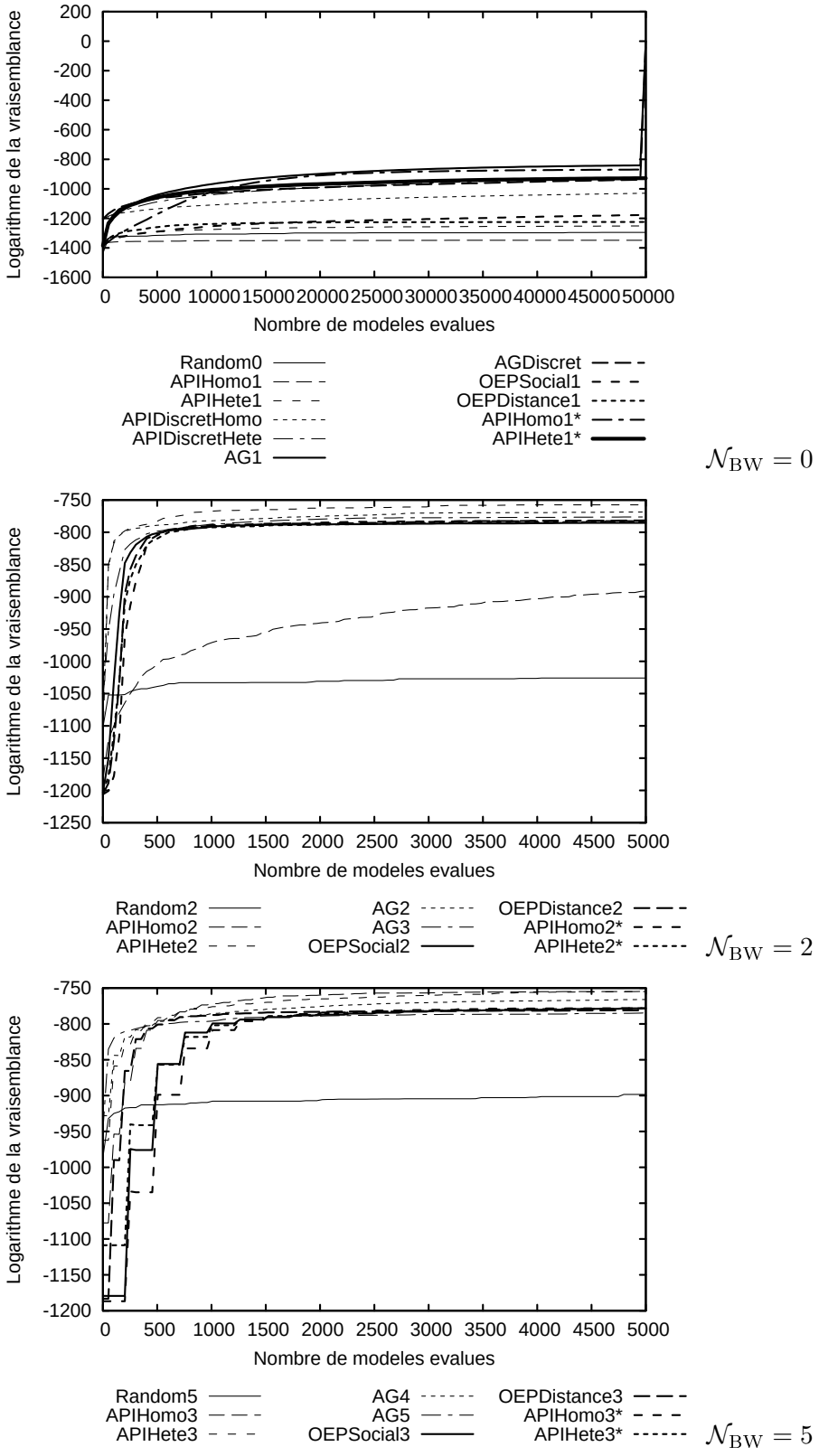
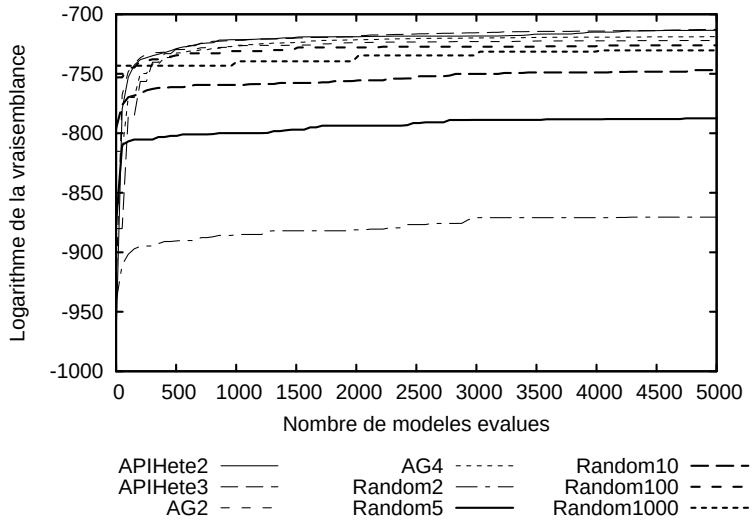
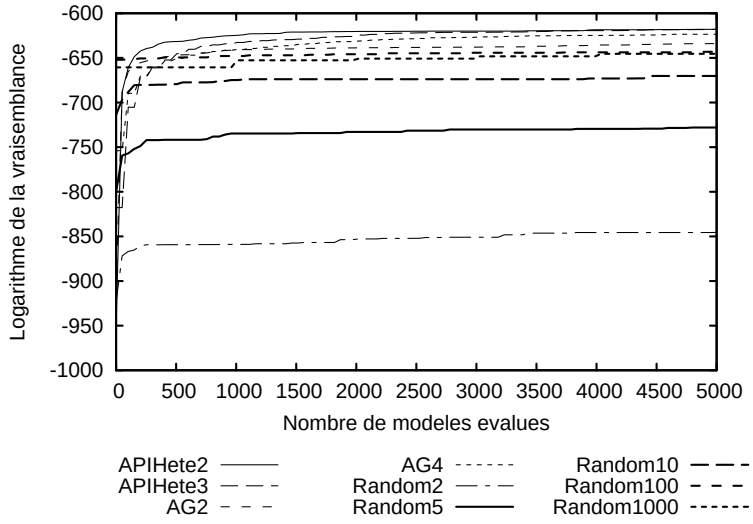


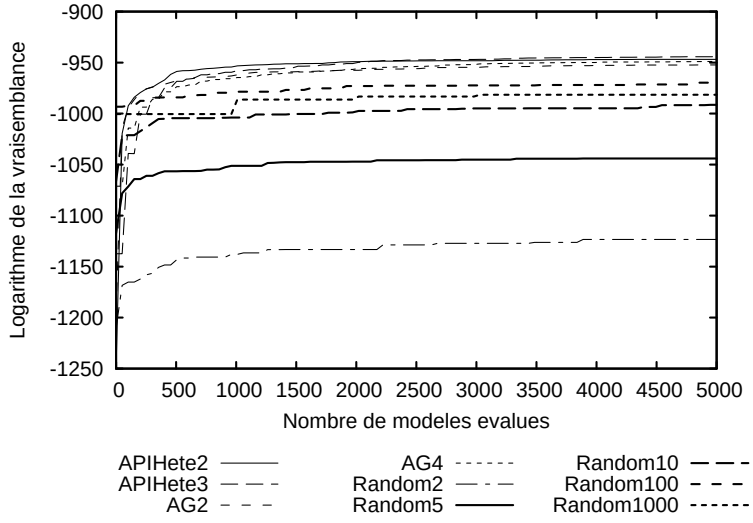
FIG. D.12 – Moyenne du logarithme de la vraisemblance pour $Img4$, $M = 32$, $N = 20$, $T = 400$.



$M = 32,\ N = 11,\ T = 400$

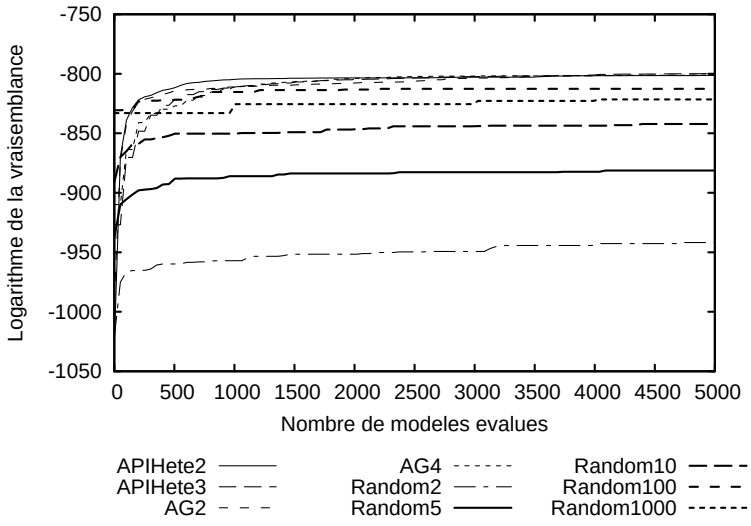


$M = 32,\ N = 20,\ T = 400$

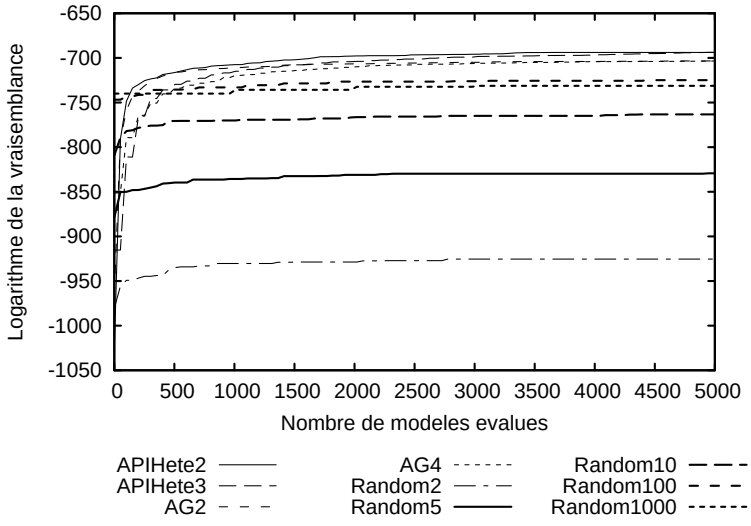


$M = 64,\ N = 11,\ T = 400$

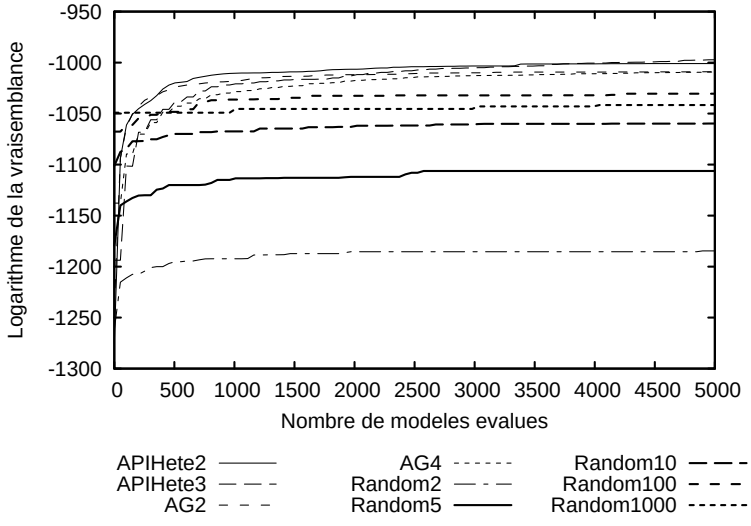
FIG. D.13 – Moyenne du logarithme de la vraisemblance pour $Img1$.



$M = 32,\ N = 11,\ T = 400$

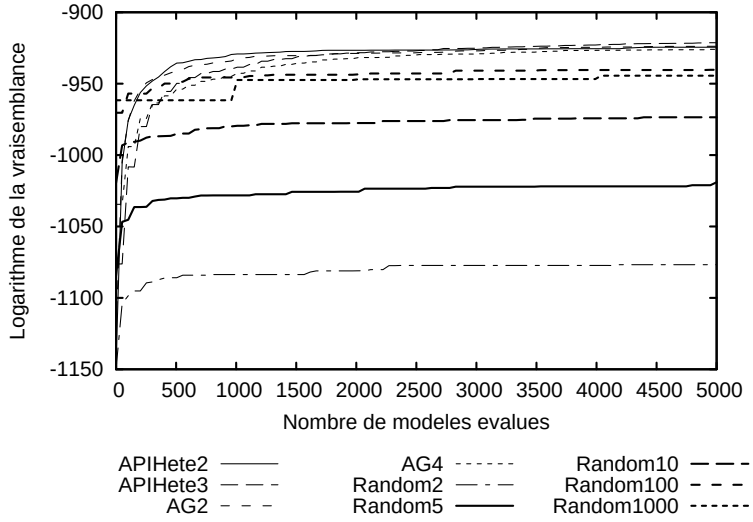


$M = 32,\ N = 20,\ T = 400$

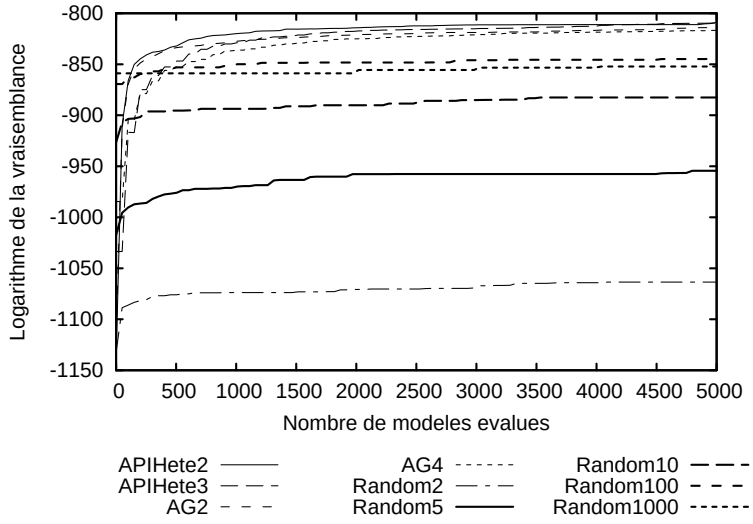


$M = 64,\ N = 11,\ T = 400$

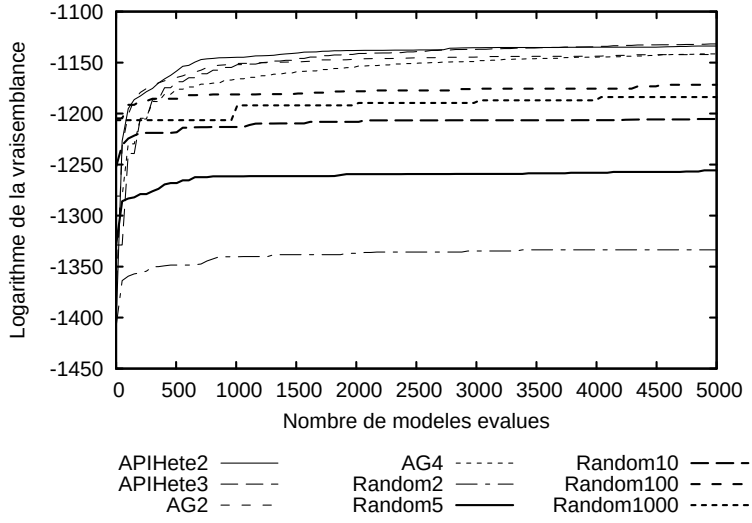
FIG. D.14 – Moyenne du logarithme de la vraisemblance pour $Img2$.



$M = 32,\ N = 11,\ T = 400$

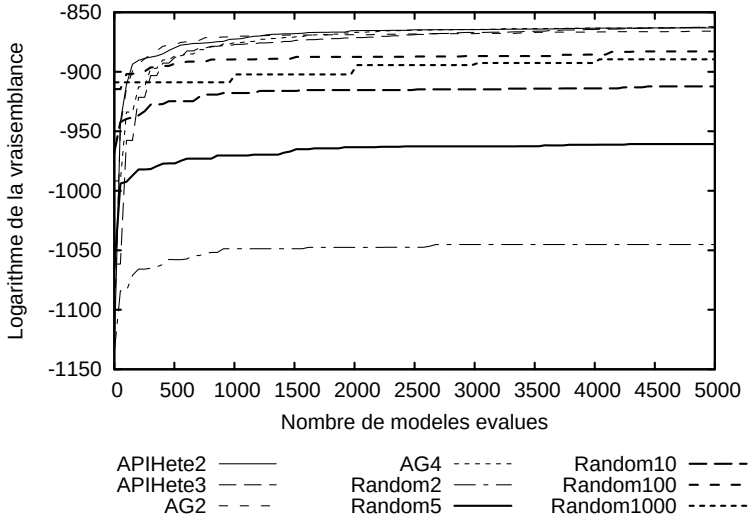


$M = 32,\ N = 20,\ T = 400$

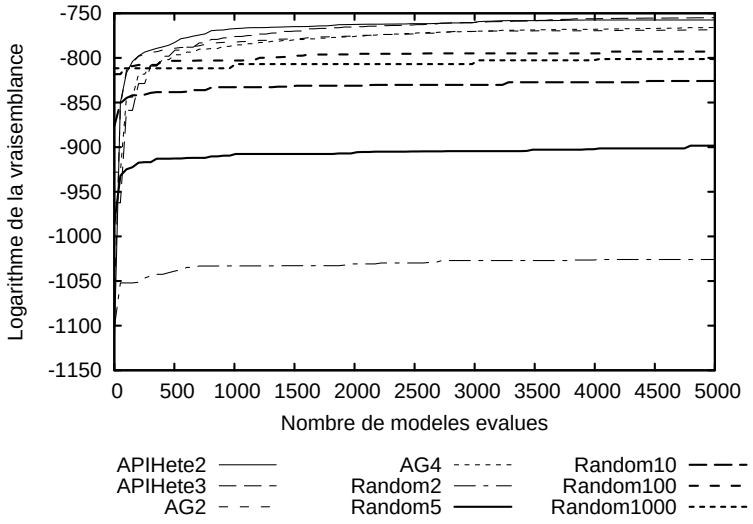


$M = 64,\ N = 11,\ T = 400$

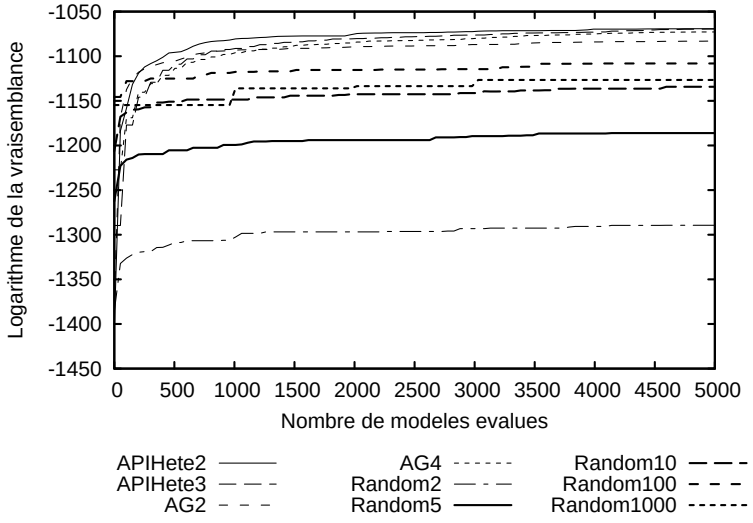
FIG. D.15 – Moyenne du logarithme de la vraisemblance pour $Img3$.



$M = 32,\ N = 11,\ T = 400$



$M = 32,\ N = 20,\ T = 400$



$M = 64,\ N = 11,\ T = 400$

FIG. D.16 – Moyenne du logarithme de la vraisemblance pour $Img4$.

Références bibliographiques

Abido, M. A. (2002). Optimal power flow using particle swarm optimization. *Electrical and Energy Systems*, 24 :563–571.

Agazzi, O. and Kuo, S.-S. (1993). Hidden Markov model based optical character recognition in the presence of deterministic transformations. *Pattern recognition*, 26(12) :1813–1826.

Amini, M.-R. (2001). *Apprentissage automatique et recherche de l'information : application à l'extraction d'information de surface et au résumé de texte*. PhD thesis, Université Paris 6.

Andrews, D. F. (1972). Plots of high dimensional data. *Biometrics*, 28 :125–136.

Ankerst, M. D. (2000). *Visual data mining*. PhD thesis, Faculty of Mathematics and Computer Science, University of Munich. Published by www.dissertation.de, ISBN : 3-89825-201-9.

Ankerst, M. D. (2001). Visual data mining with pixel-oriented visualization techniques. In *Acm sigkdd workshop on visual data mining*, San Francisco, CA.

Ankerst, M. D., Keim, A., and Kriegel, H. P. (1996). Circle segments : a technique for visually exploring large multidimensional data sets. In *Ieee visualization '96 proceedings, hot topic*, San Francisco, CA.

Aupetit, S. (2002). Indexation et recherche automatique de documents textuels à l'aide d'algorithmes fourmis et de chaînes de Markov cachés. Master's thesis, Laboratoire d'Informatique de l'Université François-Rabelais de Tours. Mémoire de DEA.

Aupetit, S., Monmarché, N., and Slimane, M. (2004a). Hybridation of symbols substitution hidden markov chains with genetic algorithms for image learning. In *Height annual meeting on health, science and technology*, Tours, France. Université François-Rabelais de Tours.

Aupetit, S., Monmarché, N., and Slimane, M. (2004b). Utilisation des chaînes de Markov cachées à substitutions de symboles pour l'apprentissage et la reconnaissance robuste d'images. In *Manifestation des jeunes chercheurs en STIC (MajecSTIC)*, Calais.

Aupetit, S., Monmarché, N., and Slimane, M. (2005a). Apprentissage de modèles de Markov cachés par essaim particulaire. In Billaut, J.-C. and Esswein, C., editors, *ROADEF'05 : 6ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*, number 1 in Collection Sciences, Technologie Informatique, pages 375–391. Presses Universitaire François-Rabelais, Tours, France.

Aupetit, S., Monmarché, N., and Slimane, M. (2005b). Fouille visuelle de dissimilarités à l'aide de matrices de scatterplots pseudo-euclidiennes. In Makarenkov, V., Cucumel, G., and Lapointe, F.-J., editors, *Comptes rendus des 12-ièmes rencontres de la Société Francophone de Classification (SFC'05)*, pages 43–46, UQAM, Quebec, Canada.

Aupetit, S., Monmarché, N., and Slimane, M. (2005c). Visualisation de dissimilarités pour des Modèles de Markov Cachés. In *6ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF'05)*, pages 70–71, Tours, France.

Aupetit, S., Monmarché, N., Slimane, M., and Liardet, S. (2005d). An Exponential Representation in the API Algorithm for Hidden Markov Models Training. In *Proceedings of the 7th International Conference on Artificial Evolution (EA'05)*, Lille, France. CD-Rom.

Aupetit, S., Slimane, M., and Monmarché, N. (2002). Les modèles de Markov cachés à substitutions de symboles. Technical Report 262, Laboratoire d'Informatique de l'Université François-Rabelais de Tours, Tours, France.

Bahl, L. R. and Jelinek, F. (1975). Decoding for channels with insertions, deletions and substitutions, with applications to speech recognition. *IEEE Transactions Theory*, 21 :404–411.

Bahl, L. R., Jelinek, F., and Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 5(2) :179–190.

Baker, J. K. (1975a). The DRAGON system-An overview. *IEEE Transactions on Acoustics, Speech, Signal Proceding*, 23(1) :24–29.

Baker, J. K. (1975b). *Stochastic Modeling as a Means of Automatic Speech Recognition*. PhD thesis, Carnegie-Mellon University.

Baker, M. P. and Wickens, C. D. (1995). Human factors in virtual environments for the visual analysis of scientific data. Technical Report ARL-95-8/PNL-95-2, NCSA-TR032 and Institute of Aviationa.

Baldi, P. and Brunak, S. (1998). *Bioinformatics : the machine learning approach*. MIT Press.

Baum, L. E. (1972). An inequality and associated maximisation technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3 :1–8.

Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of markov processes to a model for ecology. *Bull American Mathematical Society*, 73 :360–363.

Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions on manifolds. *Pac. J. Math.*, 27 :211–227.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.*, 41(1) :164–171.

Baum, L. E. and Sell, G. R. (1968). Growth transformations for functions on manifolds. *Pac. J. Math.*, 27 :211–227.

Bavoua, A. and Boujnah, T. (2005). Installation et configuration de LAM/MPI pour les grappes de calcul. Ecole Polytechnique de l'Université de Tours.

Becker, R. A. and Cleveland, W. S. (1987). Brushing Scatterplots. *Technometrics*, 29 :127–142. Reprinted in em Dynamic Graphics for Data Analysis, edited by W. S. Cleveland and M. E. McGill, Chapman and Hall, New York, 1988.

Bengio, Y. (1999). Markovian Models for Sequential Data. *Neural Computing Surveys*, 2 :129–162.

Benkirane, O. and Bouyahyaoui, M. (2003). Application des Modèles de Markov cachés à substitution de symboles (MMCSS) à la reconnaissance d'images. Ecole Polytechnique de l'Université de Tours.

Berchtold, A. (1999). The Double Chain Markov Model. *Communications in Statistics : Theory and Methods*, 28(11) :2569–2589.

Berthold, M. and Hand, D. J., editors (1998). *Intelligent data analysis : an introduction*. Springer-Verlag.

Bertier, P. and Bouroche, J.-M. (1975). *Analyse des données multidimensionnelles*. Presses Universitaires de France.

Beshers, C. and Feiner, S. (1993). AutoVisual : Rule-based design of interactive multivariate visualizations. *IEEE Computer Graphics and Applications*, 13(4) :41–49.

Beyer, H.-G. (2001). *The theory of evolution strategies.* Natural computing series. Springer.

Billingsley, P. (1961). *Statistical inference for Markov processes.* Univiversity of Chicago Press, Chicagoc.

Bilmes, J. A. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden Markov models. Technical report, University of Berkeley, ICSI-TR-97-021.

Blake, C. L. and Merz, C. J. (1998). UCI repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences. `http://www.ics.uci.edu/mlearn/MLRepository.html`.

Borg, I. and Groenen, P. (1997). *Modern multidimensional scaling : theory and applications.* Springer series in statictics. Springer.

Boughorbel, S., Tarel, J.-P., and Boujemaa, N. (2005). Conditionally positive definite kernels for svm based image recognition. In *Accepted proceedings of IEEE International Conference on Multimedia and Expo (ICME'05)*, Amsterdam, The Netherlands. `http://www-rocq.inria.fr/~tarel/icme05.html`.

Bourland, H. and Wellekens, C. (1990). Links between Markov Models and multiplayer perceptrons. *IEEE transactions on Pattern Analysis and Machine Inteligence*, 12(10) :1–4.

Brouard, T. (1999). *Algorithmes hybrides d'apprentissage de chaînes de Markov cachées : conception et applications à la reconnaissance des formes.* PhD thesis, Laboratoire d'Informatique de l'Université François-Rabelais de Tours.

Brunsdon, C., Fotheringham, A., and Charlton, M. (1998). *Case studies of visualization in the social sciences, d. unwin and p. fisher (eds.)*, technical report An Investigation of Methods for Visualising Highly Multivariate Datasets, pages 55–80. 43.

Bulaja, S. (1994). Population-based incremental learning : a method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University.

Bulaja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In Prieditis, A. and Russel, S., editors, *The International Conference on Machine Learning (ML'95)*, pages 38–46, San Mateo, CA. Morgan Kaufman Publishers.

Camiz, S. (2002). *Contribution, à partir d'exemples d'application, à la méthodologie en analyse des données.* PhD thesis, Université Paris-IX Dauphine, CEREMADE, Paris. Soutenue en juin 2002.

Cappé, O. (2001). Ten years of hmms. `http://www.tsi.enst.fr/cappe/docs/hmmbib.html`.

Cappé, O., Doucet, A., Lavielle, M., and Moulines, E. (1999). Simulation-based methods for blind maximum-likelihood filter identification. *Signal Processing*, 73 :3–25.

Card, S. K., Mackinlay, J. D., and Shneiderman, B., editors (1999). *Readings in Information Visualization : using vision to think.* Series in Interactive Technologies. Morgan Kaufmann Publishers.

Carr, D. (1995). Scanning a 4-d domain for local minima : a protein folding application. *Statistical computing and statistical graphics newsletter*, 6(2) :8–12.

CeCILL (2005). CeCILL : licence française de logiciel libre. `http://www.cecill.info/`.

Celeux, G. and Diebolt, J. (1986). L'algorithme SEM : un algorithme d'apprentissage probabiliste pour la reconnaissance des mélanges de densités. *Revue de Statistique Appliquée*, 34(2) :35–52.

Celeux, G. and Diebolt, J. (1989). Une version de type recuit simule de l'algorithme EM. Technical Report RR-1123, INRIA-Rocquencourt.

Chambers, J. M., Cleveland, W. S., Kleiner, B., and Tukey, P. A. (1983). *Graphical Methods for Data Analysis*. Chapman & Hall.

Chen, T.-Y., Mei, X.-D., Pan, J.-S., and Sun, S.-H. (2004). Optimization of hmm by the tabu search algorithm. *Journal Information Science and Engineering*, 20(5) :949–957.

Cheng, S. H. and Higham, N. J. (1998). A modified cholesky algorithm based on a symmetric indefinite factorization. *SIAM Journal on Matrix Analysis and Applications*, 19(4) :1097–1110.

Chernoff, H. (1973). The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68 :361–368.

Chesnet, S. and Herault, H. (2005). Réalisation d'un LiveCD gentoo personnalisé. Ecole Polytechnique de l'Université de Tours.

Chou, S.-Y., Lin, S.-W., and Yeh, C.-S. (1999). Cluster identification with parallel coordinates. *Pattern Recognition Letters*, 20(6) :565–572.

Clerc, M. (2005a). *L'optimisation par essaims particulaires : versions paramétriques et adaptatives*. Hermes Science - Lavoisier, Paris.

Clerc, M. (2005b). The Most Stupid Algorithm. Math stuff about PSO, `http://clerc.maurice.free.fr/pso/`.

Cleveland, W. S. (1993). *Visualizing Data*. Hobart Press, Summit, New Jersey, U.S.A, 1st edition edition.

Cox, T. F. and Cox, M. A. A. (2001). *Multidimensional scaling*. Monographs on statistics and applied probability. Chapman & Hall/CRC, Boca Raton, London, New York, Washington, D.C, second edition edition.

Darwin, C. (1859). *On the origin of species by means of natural selection or the preservation of favored races in the struggle for life*. Murray, London.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1) :1–39.

Desmarest, J. and Badille, J. (2005). Installation et configuration de MPICH2 pour les grappes de calcul. Ecole Polytechnique de l'Université de Tours.

Di Battista, G. and Eades, P. (1994). Algorithms for drawing graphs : an annoted bibliography. *Computational Geometry : Theory and Applications*, 4 :235–282.

Do, M. N. (2003). Fast approximation of Kullback-Leibler distance for dependence trees and hidden Markov models. *IEEE Signal Processing Letters*, 10(8) :250–254.

Dours, C. (1989). *Contribution à l'étude du décodage acoustico-phonétique pour la reconnaissance automatique de la parole*. PhD thesis, Université Paul Sabatier, Toulouse.

Dréo, J., Pétrowski, A., Siarry, P., and Taillard, E. (2003). *Métaheuristiques pour l'optimisation difficile*. Eyrolles, Paris.

Dugad, R. and Desai, U. B. (1996). A tutorial on hidden Markov models. Technical Report SPANN-96.1, Indian Institute of Technology, Bombay, India.

Dursteler, J. C. (2000-2003). www.infovis.net.

Eddy, S. R. (1998). Profile hidden Markov models. *Bioinformatics*, 14(9) :755–63.

Eick, S. G. (2000). Visual discovery and analysis. *IEEE Transactions on Visualization and Computer Graphics*, 6(1) :44–58.

El-Yacoubi, A., Gilloux, M., Sabourin, R., and Suen, C. Y. (1999). An HMM-based approach for off-line unconstrained handwritten word modelling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligece (PAMI)*, 21(8) :752–760.

Escofier, B. and Pagès, J. (1998). *Analyses factorielles simples et multiples : objectifs, méthodes et interprétation*. 2ème cycle - Ècoles d'ingénieurs. Dunod, Paris, 3ème édition edition.

Euler, S. and Wolf, D. (1988). Continuous hidden Markov models in speaker independent isolated word recognition. In *Proceedings of 4th EUSIPCO*, pages 1185–1188, Grenoble, France.

Falkhausen, M., Reininger, H., and Dietrich, W. (1995). Calculation of distance measures between hidden Markov models. In *Proceedings of the Eurospeech '95*, pages 1487–1490. Madrid.

Fayyad, U., Grinstein, G. G., and Wierse, A., editors (2001). *Information visualization in data mining and knowledge discovery*. Morgan Kaufmann.

Feiner, S. K. and Beshers, C. (1990). Worlds within worlds : metaphors for exploring n-dimensional virtual worlds. In *Proceedings of the 3rd annual ACM SIGGRAPH symposium on User interface software and technology*, pages 76–83, Snowbird, Utah, United States.

Feller, W. (1958). *An introduction to probability theory and its applications*, volume 1. John Wiley, New York, 2nd edition edition.

Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden markov model : Analysis and applications. *Machine Learning*, 32(1) :41–62.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems, Part II. *Annual Eugenics*, (7) :179–188.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial intelligence through simulated evolution*. Wiley.

Fonlupt, C., Robilliard, D., Preux, P., and Talbi E.-G.Fitness Landscape and Performance of Meta-Heuristics (1999). *Meta-Heuristics - Advances and Trends in Local Search Paradigms for Optimization*, chapter 18 - Fitness Landscape and Performance of Meta-Heuristics, pages 255–266. Kluwer Academic Press.

Forney Jr., G. D. (1973). The Viterbi algorithm. In *Proceedings of IEEE*, volume 61, pages 268–278.

Fua, Y. H., Ward, M. O., and Rundensteiner, E. A. (1999). Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of the conference on Visualization '99*, pages 43–50, San Francisco, California, United States. IEEE Computer Society Press Los Alamitos, CA, USA.

Furguson, J. D. (1980). Variable duration models for speech. In *Proceedings of the Symposium on the Applications of Hidden Markov Models to text and speech - IDA-CRD*, pages 8–15, Princeton NJ.

Ganapathiraju, A. (1999). Discriminative techniques in hidden Markov models. Course paper.

GHMM (2005). GHMM Library. `http://ghmm.sourceforge.net/`.

Giurgiu, M. (2002). Maximization of mutual information for training hidden markov models in speech recognition. In *3rd COST #276 Workshop*, pages 96–101, Budapest, Hungary.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, (13) :533–549.

Glover, F. (1989a). Tabu search - part I. *ORSA Journal on Computing*, pages 190–206.

Glover, F. (1989b). Tabu search - part II. *ORSA Journal on Computing*, pages 4–32.

Glover, F. and Laguna, M. (1997). *Tabu Search.* Kluwer Academic Publishers.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley.

Gower, J. C. and Legendre, P. (1986). Metric and Euclidean properties of dissimilarities coefficients. *Journal of Classification*, 3 :5–48.

Graepel, T., Herbrich, R., Bollmann-Sdorra, P., and Obermayer, K. (1999). Classification on pairwise proximity data. In *Proceedings of the 1998 conference on advances in neural information processing systems ii*, pages 438–444. MIT Press.

Grinstein, G., Pickett, R., and Williams, M. (1989). EXVIS : an exploratory visualization environment. In *Graphics Interface '89.*

Guéret, C., Monmarché, N., Slimane, M., and Cardot, H. (2003). Introduction aux Machines à Vecteur Support. Rapport Interne 273, Laboratoire d'Informatique de l'Université François-Rabelais de Tours.

Haasdonk, B. (2003). Feature space interpretation of svms with non positive definite kernels. Technical report, Computer Science Department Albert-Ludwigs, University Freiburg, 79110 Freiburg, Germany.

Hamam, Y. and Al Ani, T. (1996). Simulated annealing approach for Hidden Markov Models. In *4th WG-7.6 Working Conference on Optimization-Based Computer-Aided Modeling and Design, ESIEE, France.*

Harris, R. J. (1985). *A primer of multivariate statistics. second edition.* Academic press, inc.

Hartley, H. O. (1958). Maximum likelihood estimation from incomplete data. *Biometrics*, 14 :174–194.

Hauser, H., Ledermann, F., and Doleisch, H. (2002). Angular brushing of extended parallel coordinates. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, pages 127–130. IEEE Computer Society, Washington, DC, USA.

Herman, I., Melançon, G., and Marshall, M. S. (2000). Graph Visualization and Navigation in Information Visualization : A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1) :24–43.

Hertz, A., Taillard, E., and de Werra, D. (1992). A Tutorial on tabu search. In *Proceedings of Giornate di Lavoro AIRO'95 (Enterprise Systems : Management of Technological and Organizational Changes*, pages 13–24.

Hoffman, P., Grinstein, G., Marx, K., Grosse, I., and Stanley, E. (1997). DNA Visual And Analytic Data Mining. In Yagel, R. and Hagen, H., editors, *Proceedings of the 8th conference on Visualization '97*, pages 437–442, Phoenix, Arizona, United States.

Hoffman, P., Grinstein, G., and Pinkney, D. (1999). Dimensional anchors : a graphic primitive for multidimensional multivariate information visualizations. In *Proceedings of the 1999 workshop on new paradigms in information visualization and manipulation*, pages 9–16, Kansas City, Missouri, United States. ACM Press, New York, NY, USA.

Hoffmann, P. and Grinstein, G. (2005). Visualizations for High Dimensional Data Mining - Table Visualizations. `http://home.comcast.net/~patrick.hoffman/viz/MIV-datamining.pdf`.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* University of Michigan Press : Ann Arbor, MI.

HTK (2005). HTK Speech Recognition Toolkit. `http://htk.eng.cam.ac.uk/`.

Hu, X., Eberhart, R. C., and Shi, Y. (2003). Swarm intelligence for permutation optimization : a case-study of n-queens problems. In *Proceedings of the ieee swarm intelligence symposium*, pages 243–246, Indianapolis.

Hublier, A. and Téboul, V. (2003). Tests sur l'utilisation de CMCSS dans la reconnaissance d'images. Ecole Polytechnique de l'Université de Tours.

Inselberg, A. (2002). Visualization and data mining of high-dimensional data. *Chemometrics and Intelligent Laboratory Systems*, 60 :147–159.

Inselberg, A. and Dimsdale, B. (1990). Parallel coordinates : a tool for visualizing multi-dimensional geometry. In Kaufman, A., Rosenblum, L., and Nielson, G. M., editors, *Proceedings of the 1st conference on Visualization '90*, pages 361–378, San Francisco, California. IEEE Computer Society Press Los Alamitos, CA, USA.

Java (2005). Java technology. `http://java.sun.com/`.

Jelinek, F., Bahl, L. R., and Mercer, L. (1975). Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, 21(3).

Johnson, B. and Shneiderman, B. (1991). Tree-maps : A space-filling approach to the visualization of hierarchical information structures. In *Proceedings of IEEE Visualization '91*, pages 284–291.

Juang, B. H. and Rabiner, L. R. (1986). Mixture autoregressive hidden Markov models for speaker independent isolated word recognition. In *IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP'86)*, pages 41–44, Tokyo.

Juang, B.-H. and Rabiner, L. R. (1990). The segmental k-means algorithm for estimating parameters of hidden Markov models. *IEEE transactions on acoustics, speech and signal processing*, 38(9) :1639–1641.

Kapadia, S. (1998). *Discriminative training of hidden Markov models*. PhD thesis, Downing College, University of Cambridge.

Keim, D. A. (1997). Visual techniques for exploring databases. Invited Tutorial, International Conference on Knowledge Discovery in Databases (KDD'97), Newport Beach, CA, USA.

Keim, D. A. and Kriegel, H. P. (1994). VisDB : Database Exploration using Multidimensional Visualization. *Computer Graphics & Applications Journal*.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the ieee international joint conference on neural networks*, volume 4, pages 1942–1948. IEEE.

Kennedy, J. and Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Proceedings of Conference on Systems, Man, and Cybernetics*, pages 4104–4109, Piscataway, NJ.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimizing by simulated annealing. *Science*, 220(4598) :671–680.

Kriouile, A. (1990). *La reconnaissance automatique de la parole et les modèles markoviens cachés*. PhD thesis, Université Nancy I.

Kundu, A. and Bahl, P. (1988). Recognition of handwritten script : a hidden Markov model based approach. In *International Conference on Acoustics, Speech, Signal Processing (ICASSP'88)*, pages 928–931.

Kwong, S. and Chau, C. (1997). Analysis of Parallel Genetic Algorithms on HMM Based Speech Recognition System. In *Proceedings of ICASSP'97*, pages 1229–1233.

LAM/MPI (2005). LAM/MPI Parallel Computing. `http://www.lam-mpi.org/`.

Ledermann, F. (2002). Parvis. `http://home.subnet.at/flo/mv/parvis/`.

Lesné, O. and Mahnich, I. (2003). Projet ALICES : Explications et outils. Ecole Polytechnique de l'Université de Tours.

Lesné, O. and Perotin, M. (2004a). Chaînes de Markov cachées à substitution de symboles. Ecole Polytechnique de l'Université de Tours.

Lesné, O. and Perotin, M. (2004b). Installation d'OpenMosix en salle Unix. Ecole Polytechnique de l'Université de Tours.

Liporace, L. R. (1982). Maximum likelihouod estimation for multivariate observatiuons of Markov sources. *IEEE Transactions on Information Theory*, 28 :729–734.

Ljolje, A., Ephraim, Y., and Rabiner, L. R. (1990). Estimation of hidden Markov model parameters by minimizing empirical error rate. In *IEEE International Confonference on Acoustic, Speech, Signal Processing*, pages 709–712, Albuquerque.

Locatelli, M. (2002). *Handbook of Global Optimization*, volume 2, chapter Simulated annealing algorithms for continuous global optimization, pages 179–229. Kluwer Academic Publishers.

Mari, J.-F., Le Ber, F., and Benoit, M. (2000). Fouille de données agricoles par Modèles de Markov cachés. In *Journées francophones d'Ingénierie des Connaissances - IC'2000*, pages 197–205, Toulouse, France.

Markov, A. A. (1913). An example of statistical investigation in the text of "Eugene onyegin" illustrating coupling of "tests" in chains. In *Proceedings of Academic Scienctific St. Petersburg*, VI, pages 153–162.

Martin, A. R. and Matthew O. W. (1995). High Dimensional Brushing for Interactive Exploration of Multivariate Data. In *IEEE Conference on Visualization '95*.

Martin, L. (2005). Détection d'intrusions basée sur les chaînes de Markov cachées à partir de flux réseaux. Ecole Polytechnique de l'Université de Tours.

Martin, L. and Pérotin, M. (2003). ALICES : Mise en place d'une grille de calcul sous openmosix. Ecole Polytechnique de l'Université de Tours.

Maxwell, B. and Anderson, S. (1999). Training hidden markov models using population-based learning. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation COonference (GECCO'99)*, volume 1, page 944, Orlando, Florida, USA. Morgan Kaufmann.

Michalewicz, Z. (1992). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag, third, revised and extended edition edition.

Monmarché, N. (2000). *Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation*. Thèse de doctorat, Laboratoire d'Informatique de l'Université François-Rabelais de Tours.

Monmarché, N. (2002). Algorithmes à base de fourmis artificielles pour les problèmes à variables continues. In *Actes des quatrièmes journées nationales de la ROADEF*, pages 224–225, Paris, France.

Monmarché, N., Venturini, G., and Slimane, M. (2000). On how it Pachycondyla apicalis ants suggest a new search algorithm. *Future Generation Computer Systems*, 16(8) :937–946.

Monvoisin, B. and Viaux, C. (2005). Installation et configuration de LAM/MPI pour les grappes de calcul. Ecole Polytechnique de l'Université de Tours.

Moon, T. K. (1996). The Expectation Maximization algorithm. *IEEE signal processing magazine*, pages 47–60.

MPI (2005). The Message Passing Interface (MPI) standard. `http://www-unix.mcs.anl.gov/mpi/`.

Mroz, L., Loffelmann, H., and Groller, E. (1998). Selected trends in scientific visualization. Technical Report TR-186-2-98-15, TECHNISCHE UNIVERSITAT WIEN, Institut fur Computergraphik und Algorithmen, Abteilung fur Computergraphik.

Navillat, D. and Marty, P. (2005). Etudes des implémentations MPI-2. Ecole Polytechnique de l'Université de Tours.

NetKit (2005). Netkit's Remote Shell Suite. `ftp://ftp.uk.linux.org/pub/linux/Networking/netkit/`.

Newman, D. J., Hettich, S., Blake, C. L., and Merz, C. J. (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. University of California, Irvine, Dept. of Information and Computer Sciences.

Olivier, C., Avila, M., Courtellemont, P., Paquet, T., and Lecourtier, Y. (1993). Handwritten word recognition by image segmentation and hidden Markov model. In *Proceedings of IEEE IES 3*, pages 2093–2097.

Omran, M., Salman, A., and Engelbrecht, A. (2002). Image classification using particle swarm optimization. In *4th asia-pacific conference on simulated evolution and learning.*

Ong, C., Mary, X., Canu, S., and Smola, A. (2004). Learning with non-positive kernels. In *Proceedings of the 21st international conference on machine learning.*

OpenMosix (2005). OpenMosix : an open source linux cluster project. `http://openmosix.sourceforge.net/`.

OpenSSH (2005). OpenSSH. `http://www.openssh.com/`.

Paquet, U. and Engelbrecht, A. (2003). Training support vector machines with particle swarms. In *International joint conference on neural networks*, Portland, OR.

Paul, D. B. (1985). Training of HMM recognizers by simulated annealing. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 13–16.

Peiti, S. (2005). Segmentation d'images à l'aide de MMCSS. Ecole Polytechnique de l'Université de Tours.

Pekalska, E., Paclik, P., and Duin, R. P. W. (2002). A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, 2 :175–211.

Perl (2005). The perl directory. `http://www.perl.org/`.

Petrie, T. (1969). Probabilistic functions of finite state Markov chains. *Ann. Math. Stat*, 40 :97–115.

Pickett, R. M. and Grinstein, G. G. (1988). Iconographics display for visualizing multidimensional data. In *Proceedings IEEE Conference on Systems, Man, and Cybernetics*, pages 514–519.

Pieczynski, W. (1990). Mixture of distributions, Markov random fields and unsupervised Bayesian segmentation of images. Technical Report 122, L.S.T.A., Universite Paris VI, Paris, France.

Pieczynski, W. (2003). Arbres de Markov Triplet et fusion de Dempster-Shafer. *Comptes Rendus de l'Académie des Sciences - Mathématique*, 336(10) :869–872.

Poritz, A. B. and Richter, A. G. (1986). On hidden Markov models in isolated word recognition. In *IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP'86)*, pages 705–708, Tokyo.

PVM (2005). PVM : Parallel Virtual Machine. `http://www.csm.ornl.gov/pvm/pvm\_home.html`.

Python (2005). Python programming language. `http://www.python.org/`.

Quinlan, J. R. (1993). *C4.5 : Programs for Machine Learning.* Morgan Kauffman.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286.

Rabiner, L. R. and Levinson, S. E. (1985). A speaker independent, syntax directed connected word recognition system based on hidden Markov models and level building. *IEEE Transactions on Acoustics, Speech, Signal Processing*, 33(3) :561–573.

Rabiner, L. R., Levinson, S. E., and Sondhi, M. M. (1983). On the application of vector quantization and hidden Markov models to speaker-independant isolated word recognition. *The Bell System Technical Journal*, 62 :1075–1105.

Rameshkumar, K., Suresh, R. K., and Mohanasundaram, K. M. (2005). Discrete particle swarm optimization (dpso) algorithm for permutation flowshop scheduling to minimize makespan. In Wang, L., Chen, K., and Ong, Y.-S., editors, *ICNC (3)*, volume 3612 of *Lecture Notes in Computer Science*, pages 572–581. Springer.

Rasmussen, T. K. and T. Krink (2003). Improved hidden Markov model training for multiple sequence alignment by a particle swarm optimization - evolutionary algorithm hybrid. *BioSystems*, (72) :5–17.

Ratnaweera, A., Halgamuge, S. K., and Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 8(3) :240–255.

Robertson, G., Mackinlay, J., and Card, S. (1991). Cone Trees : Animated 3D Visualizations of Hierarchical Information. In Robertson, S. P., Olson, G. M., and Olson, J. S., editors, *Proceedings of the SIGCHI conference on Human factors in computing systems : Reaching through technology*, pages 189–194, New Orleans, Louisiana, United States.

Rosemberg, A. E. and Colla, A. M. (1987). A connected speech recognition system based on spotting diphone-like segments-preliminary results. In *Proceedings of IEEE International Confonference on Acoustics, Speech, Signal Processing (ICASSP'87)*, pages 85–87, Dallas.

Salman, A., Ahmad, I., and Al-Madani, S. (2003). Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems*, 26 :363–371.

Samaria, F. and Harter, A. (1994). Parameterisation of a stochastic model for human face identification. In *IEEE workshop on Applications of Computer Vision*, Florida.

Saul, L. and Rahim, M. (2000). Maximum likelihood and minimum classification error factor analysis for automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 8(2) :115–125.

Schluter, R., Macherey, W., Kanthak, S., Ney, H., and Welling, L. (1997). Comparison of optimization methods for discriminative training criteria. In *EUROSPEECH '97, 5th European Conference on Speech Communication and Technology*, pages 15–18, Rhodes, Greece.

Schnabel, R. B. and Eskow, E. (1999). A revised modified cholesky factorization algorithm. *SIAM Journal on Optimization*, 9(4) :1135–1148.

Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors (1999). *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA.

Scullin, W. H., Kwan, T. T., and Reed, D. A. (1995). Real-time Visualization of World Wide Web Traffic. In *Symposium on Visualizing Time-Varying Data*.

Serradura, L., Vincent, N., and Slimane, M. (2001). Modélisation, par des chaînes de Markov cachées, des thèmes de textes formatés. In *4e Colloque International sur le Document Electronique (CIDE)*, pages 297–301, Toulouse, France.

Shannon, C. C. (1948). A mathematical theory of communications. *Bell System Techology Journal*, 27 :379–423, 623–656.

Shannon, C. C. (1951). Prediction and entropy of printed English. *Bell System Techology Journal*, 30 :50–64.

Siegel, J., Farrell, E., Goldwyn, R., and Friedman, H. (1972). The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery*, (72) :126–141.

Slimane, M. (2002). Les chaînes de Markov cachées : définitions, algorithmes, architectures. Rapport interne n°260, Université François-Rabelais de Tours, Laboratoire d'Informatique, Tours, France.

Slimane, M., Brouard, T., Venturini, G., and Asselin de Beauville, J.-P. (1999). Apprentissage non-supervisé d'images par hybridation génétique d'une chaîne de Markov cachée. *Traitement du signal*, 16(6) :461–475.

Soukhal, A., Kelarestaghi, M., Slimane, M., and Martineau, P. (2001a). Application des chaînes de Markov cachées au problème d'ordonnancement dans une cellule robotisée. In *Conférence Internationale sur la productique (CIP'01)*, pages 151–156, Algérie.

Soukhal, A., Kelarestaghi, M., Slimane, M., and Martineau, P. (2001b). Hidden Markov Models and scheduling problem with transportation consideration. In *15th Annual European Simulation Multiconference (ESM'01)*, pages 836–840, Prague (Tchéquie).

Tasgetiren, M. F., Sevkli, M., Liang, Y.-C., and Gencyilmaz, G. (2004). Particle swarm optimization algorithm for permutation flowshop. In Dorigo, M., Birattari, M., Blum, C., Gambardella, L., Mondada, F., and Stützle, T., editors, *Proceedings of ants 2004 – fourth international workshop on ant colony optimization and swarm intelligence*, volume 3172 of *Lecture Notes in Computer Science*, Brussels, Belgium. Springer Verlag.

Thomsen, R. (2002). Evolving the topology of hidden Markov models using evolutionary algorithms. In *Proceedings of Parallel Problem Solving from Nature VII (PPSN-2002)*, pages 861–870.

Thomy, J. M. (2003). Visualisation 3D d'une classification en réalité virtuelle. Ecole Polytechnique de l'Université de Tours. Projet de Fin d'Etudes.

Torgeson, W. S. (1965). Multidimensional scaling of similarity. *Psychometrika*, 30 :379–393.

van der Merwe, D. W. and Engelbrecht, A. P. (2003). Data clustering using particle swarm optimization. In *Ieee congress on evolutionary computation*, pages 215–220, Canberra, Australia.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer-Verlag, New York.

Vertanen, K. (2004). An overview of discriminative training for speech recognition. Technical report, University of Cambridge.

Vesanto, J. (1997). Data mining techniques based on the self-organizing map. Master's thesis, Helsinki University of Technology, Espoo, Finland.

Vesanto, J. (1999). Som-based data visualization methodsu. *Intelligent Data Analysis*, 3(2) :111–126.

Vesanto, J., Himberg, J., Siponen, M., and Simula, O. (1998). Enhancing som based data visualizationW. In *Proceedings of the international conference on soft computing and information/intelligent systems (iizuka'98)*, pages 64–67, Iizuka, Japan.

Vidal, J. (2004). Détection d'évènements anormaux et de changements de comportements dans des graphiques de flux réseaux. Ecole Polytechnique de l'Université de Tours.

Vihola, M., Harju, M., Salmela, P., Suontausta, J., and Savela, J. (2002). Two dissimilarity measures for HMMs and their application in phoneme model clustering. In *Proceedings ICASSP 2002, 2002 IEEE International Conference on Acoustics Speech and Signal Processing*, pages 933–936, Orlando, Florida, USA.

Viterbi, A. J. (1967). Error bounds for convolutionnal codes and asymptotically optimum decoding algorithm. *IEEE transactions on information theory*, 13 :260–269.

Wang, J., Yu, B., and Gasser, L. (2002a). Classification visualization with shaded similarity matrix. Technical report, Graduate school of library and information science, University of Illinois at Urbana-Champaign.

Wang, J., Yu, B., and Gasser, L. (2002b). Concept tree based clustering visualization with shaded similarity matrices. In Kumar, V., Tsumoto, S., Zhong, N., Yu, P., and Wu, X., editors, *Proceedings of 2002 ieee international conference on data mining*, pages 697–700, Maebashi, Japan. IEEE Computer Society.

Ward, M. O. and Keim, D. (1997). Screen layout methods for multidimensional visualization. In *Proceedings of codata euro-american workshop on visualization of information and data*, page 2p.

Ward, M. O., LeBlanc, J. T., and Tipnis, R. (1994). N-land : a graphical tool for exploring n-dimensional data. In *Proceedings of Computer Graphics International Conference'94*, Melbourne, Australia.

Wijk, J. v. and van Liere, R. (1993). Hyperslice - visualization of scalar functions of many variables. In Press, I. C. S., editor, *Proceedings visualization'93*, Los Alamitos, CA.

Wills, G. J. (1999). Nicheworks - interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics*, 8(2) :190–212.

Wiseman, J. (1998). Chernoff Faces (in Java). `http://people.cs.uchicago.edu/~wiseman/chernoff/`.

Wong, P. C. and Bergeron, R. D. (1997). 30 Years of Multidimensional Multivariate Visualization. *Scientific Visualization, Overviews, Methodologies, and Techniques*, pages 3–33. IEEE Computer Society, Washington, DC, USA,.

Zaragoza, H. and Gallinari, P. (1998). Modèle hiérarchique et extraction de l'information et d'extraction de l'information textuelle de surface. In *Journée Francophones d'apprentissage (JFA'98)*, page 12.

Liste des figures

Liste des tableaux

Listes des algorithmes

Résumé :

Dans ce travail de thèse, nous présentons plusieurs contributions visant à améliorer l'utilisation des modèles de Markov cachés (MMC) dans les systèmes d'intelligence artificielle. Nous nous sommes concentrés sur trois objectifs : l'amélioration de l'apprentissage de MMC, l'expérimentation d'un nouveau type de MMC et la visualisation de dissimilarité pour mieux comprendre les interactions entre MMC. Dans la première partie, nous proposons, évaluons et comparons plusieurs nouvelles applications de métaheuristiques biomimétiques classiques (les algorithmes génétiques, l'algorithme de fourmis artificielles API et l'optimisation par essaim particulaire) au problème de l'apprentissage de MMC. Dans la deuxième partie, nous proposons un nouveau type de modèle de Markov caché, appelé modèle Markov caché à substitutions de symboles (MMCSS). Un MMCSS permet d'incorporer des connaissances *a priori* dans le processus d'apprentissage et de reconnaissance. Les premières expérimentations de ces modèles sur des images démontrent leur intérêt. Dans la troisième partie, nous proposons une nouvelle méthode de représentation de dissimilarité appelée matrice de *scatterplots* pseudo-euclidienne (MSPE), permettant de mieux comprendre les interactions entre des MMC. Cette MSPE est construite à partir d'une technique que nous nommons analyse en composantes principales à noyau indéfini (ACPNI). Nous terminons par la présentation de la bibliothèque HMMTK, développée au cours de ce travail. Cette dernière intègre des mécanismes de parallélisation et les algorithmes développés au cours de la thèse.

Mots clés :

modèles de Markov cachés, apprentissage, optimisation, métaheuristique biomimétique, algorithme génétique, algorithme de fourmis artificielles API, optimisation par essaim particulaire, incorporation de connaissances, substitution de symboles, matrice de *scatterplots*, espace pseudo-euclidien, analyse en composantes principales, méthode à noyau

Abstract :

In this PhD thesis, we present many contributions aimed at the improvement on the utilization of hidden Markov Models (HMMs) in artificial intelligence systems. We considered three main objectives : improving HMM training, experimenting a new HMM and visualizing interaction among HMMs. In the first part, we propose, evaluate and compare many new adaptations of classic biomimetic metaheuristics (genetic algorithms, API artificial ants algorithm, particle swarm optimization) applied to the problem of HMMs training. In the second part, we propose a new kind of HMM which we named symbols substitution hidden Markov models (SSHMMs). A SSHMM allows an expert to incorporate *a priori* knowledge in the training and the recognition tasks. First experiments with such models show that SSHMMs are of great interests at least for images learning and recognition. In the third part, we propose a new visualization technique to tackle dissimilarity. This technique, which we named the pseudo-euclidean scatterplots matrix (PESM), allows a better understanding of interaction between HMMs. This PESM is built from techniques which we named indefinite kernel principal component analysis (IKPCA). Finally, our research concludes with the description of the HMMTK software library developed along this work. The library integrates parallelization mechanisms and algorithms developed during the thesis.

Key words :

hidden Markov models, training, optimization, biomimetic metaheuristic, genetic algorithm, API artificial ants algorithm, particle swarm optimization, knowledges incorporation, symbol substitution, scatterplots matrix, pseudo-euclidean space, principal component analysis, kernel method

---

Université François-Rabelais de Tours, Laboratoire d'Informatique, UPRES-EA 2101, Équipe Handicap et Nouvelles Technologies, Polytech'Tours, Département Informatique, 64 avenue Jean Portalis, 37200 Tours (`http://www.li.univ-tours.fr/`, `http://www.polytech.univ-tours.fr/`).